Calcul incrémental des groupes d'homologie d'un objet au cours d'un processus de construction

Wassim Rharbaoui

Thèse de Doctorat

Pour l'obtention du grade de :

Docteur de l'Université de Poitiers

Faculté des Sciences Fondamentales et Appliquées
(Diplôme national - arrêté du 25 mai 2016)

École doctorale : Sciences et Ingénierie des Systèmes, Mathématiques, Informatique
Secteur de recherche : Informatique

****

Calcul incrémental des groupes d'homologie d'un objet au cours d'un processus de construction

****

Présentée par :
Wassim RHARBAOUI

Composition du jury :

Dominique BECHMANN ..... Université de Strasbourg, Professeur
Alexandra BAC ..... Université d'Aix-Marseille, Maître de conférences HDR
Steve OUDOT ..... Inria Saclay, Directeur de recherche
Pascal LIENHARDT ..... Université de Poitiers, Professeur
Sylvie ALAYRANGUES ..... Université de Poitiers, Maître de conférences
Samuel PELTIER ..... Université de Poitiers, Maître de conférences

À ma famille.

Remerciements

En premier lieu, je tiens à remercier mon directeur de thèse Pascal LIENHARDT et mes encadrants Sylvie ALAYRANGUES et Samuel PELTIER pour leur soutien sans faille pendant toutes ces années. Merci de m'avoir accompagné dans ma découverte du monde de la recherche.

Je remercie Guillaume DAMIAND d'avoir accepté de partager avec moi une part de son expérience au travers d'une collaboration, et avec qui j'ai pris grand plaisir à travailler.

Merci aux membres du jury, Dominique BECHMANN, Alexandra BAC et Steve OUDOT pour leur travail de relecture de ce manuscrit et d'évaluation de mes travaux.

Merci à tous les collègues avec qui j'ai pu travailler et échanger au fil de ces années, me faisant profiter d'une part de leur vécu. En particulier, merci à Hakim BELHAOUARI, Thierry URRUTY, Sébastien HORNA et Gaëlle LARGETEAU.

J'aimerais également remercier ceux dont le travail a contribué de manière plus ou moins directe au bon déroulement de ma thèse. En particulier, merci à Sophie HARDOUIN, Caroline COURBIER, Alicia LECESVE, Sylvie DUCLAUD, Yoann SIMON et Bruno MERCIER.

Enfin, merci à tous mes camarades du quotidien grâce à qui j'ai pu me changer les idées lorsqu'il le fallait. Pour cela, je remercie chaleureusement Simon COURTIN, Arthur CAVALIER, Valentin GAUTHIER, Lydie RICHAUME, Hermine CHATOUX, Elsa TAMISIER, Paul DEQUIDT, Angélique PERRILLAT et Boubacar DIALLO.

Résumé

En modélisation géométrique à base topologique, les objets manipulés sont subdivisés en cellules de différentes dimensions (sommets, arêtes, faces, volumes...). Dans ce cadre, le calcul d'invariants topologiques (orientabilité, contractilité, caractéristique d'Euler...) permet de caractériser la structure de ces objets. En particulier, l'homologie est un invariant topologique usuellement étudié, permettant intuitivement de caractériser les trous d'un objet en toute dimension (composantes connexes en dimension 0, tunnels en dimension 1, cavités en dimension 2 etc...).

Classiquement, le calcul de l'homologie d'un objet nécessite d'étudier les relations d'incidence de toutes ses cellules. Dans cette thèse, on s'intéresse au calcul incrémental des variations de l'homologie d'un objet évoluant dans un processus de construction. Pour cela, nous utilisons des résultats de *l'homologie effective* [1], et plus particulièrement le *théorème des suites exactes courtes effectives* (théorème SECE). Le passage d'une étape de construction à l'autre se fait par application d'une opération *locale* consistant à fusionner des cellules (identification), ou, à l'inverse, à les scinder (désidentification). Le théorème SECE est utilisé pour maintenir une *équivalence homologique* au fil des étapes. Cette dernière associe l'objet à un plus petit objet de "même" homologie. À chaque étape, l'homologie peut être calculée à partir du petit objet, ce qui est plus efficace que de la calculer à partir de l'objet lui-même.

Dans ce contexte, nous proposons une analyse du coût des calculs mis en jeu par le théorème SECE. Il en résulte que, pour calculer les rangs des groupes d'homologie à chaque étape, la complexité en temps du maintien de l'équivalence homologique dépend seulement du nombre de cellules impactées par l'opération (et de leur étoile), et la complexité en espace croît en fonction du nombre de cellules impactées par l'opération. Pour garantir ces complexités en pratique, nous distinguons plusieurs prérequis qu'une implémentation doit respecter. Nous proposons une structure de données vérifiant ces prérequis. Elle inclut des informations pour suivre l'évolution des cellules d'une structure topologique au fil du processus de construction, c'est-à-dire le fait que des cellules puissent mourir ou être créées à chaque étape. En fonction de ces évolutions, elle est utilisée pour mettre à jour les éléments maintenus au fil du processus et utilisés dans le théorème SECE, comme par exemple des matrices de bord de *complexes de chaînes*.

Ensuite, nous nous intéressons aux cas où l'objet est trop volumineux pour être manipulé par une seule unité de calcul. Nous proposons un algorithme permettant de calculer l'homologie d'un objet distribué évoluant dans un processus de construction composé uniquement d'identifications. L'objet est manipulé implicitement au travers de sa distribution et d'une identification permettant de le reconstruire à partir de sa distribution. À chaque étape, ces données sont mises à jour afin de permettre le calcul de l'homologie de l'étape suivante sans avoir à reconstruire l'objet.

Enfin, nous mettons en évidence un cadre commun à l'*homologie effective* et à l'*homologie persistante*. En particulier, nous nous intéressons aux travaux concernant les *tours*, des séquences de complexes simpliciaux reliés entre eux par des applications simpliciales [2].

[1] Julio Rubio and Francis Sergeraert. Constructive homological algebra and applications. Technical report, Universidad de la Rioja, Université Grenoble Alpes, 2006

[2] Tamal K Dey, Fengtao Fan, and Yusu Wang. Computing topological persistence for simplicial maps. In *Proceedings of the thirtieth annual symposium on Computational geometry*, pages 345–354, 2014

Mots-clefs : Modélisation géométrique, Homologie, Suites exactes courtes effectives, Structures topologiques, Complexes de chaînes, Parallélisme.

Abstract

Topology-based geometric modeling involves objects subdivided into cells of various dimensions (vertices, edges, faces, volumes...). Computing topological invariants (orientability, contractibility, Euler characteristic...) helps to characterize the structure of these objects. Especially, homology is a topological invariant usually studied to characterize the holes of an object in any dimension (connected components in dimension 0, tunnels in dimension 1, cavities in dimension 2 etc...).

Classically, computing the homology of an object requires to study the incidence relationships between all of its cells. In this thesis, we're interested in computing incrementally the homology variations of an object evolving during a construction process. To achieve this goal, we are using *effective homology* results [1], and more specifically the effective *short exact sequences theorem* (SES theorem).

Progressing from a construction step to an other is done by applying a *local* operation on the object and consists either in merging cells (identification) or in splitting cells (desidentification). The SES theorem is used to maintain a *homological equivalence* over the construction process. It relates the object to a smaller one having "the same" homology. At each step, homology is computed using this small object rather than the object itself, so that the computation is more efficient.

In this context, we analyze the computational costs implied by the SES theorem. We observe that, in order to compute homology ranks at each step, the time complexity to maintain a homological equivalence depends on the amount of cells that the operation impacts (and their star), and the space complexity between two steps also grows accordingly. To ensure these results in practice, we highlight some prerequisites that an implementation has to fulfill. We provide a data structure that can be used for this purpose. It consists in maintaining some data in order to track the evolution of the cells of a topological structure over the construction process, that is the fact that some cells may die or be created. Depending on these evolutions, it is used to update every elements maintained over the construction process and used by the SES theorem, such as the boundary matrices of *chain complexes*.

Next, we suppose that the object is too bulky to be manipulated by a single computation unit. We propose an algorithm to compute the homology of a distributed object evolving during a construction process made up only of identifications. The object is implicitly represented through its distribution and an identification describing a way to reconstruct it from its distribution. At each step, when an identification is applied on the object, these data are updated so that, at the next step, homology can be computed without reconstructing the object.

At last, we provide a common framework between *effective homology* and *persistent homology*. More precisely, we look into the works dealing with towers, which are sequences of simplicial complexes linked to one another through simplicial maps [2].

[1] Julio Rubio and Francis Sergeraert. Constructive homological algebra and applications. Technical report, Universidad de la Rioja, Université Grenoble Alpes, 2006

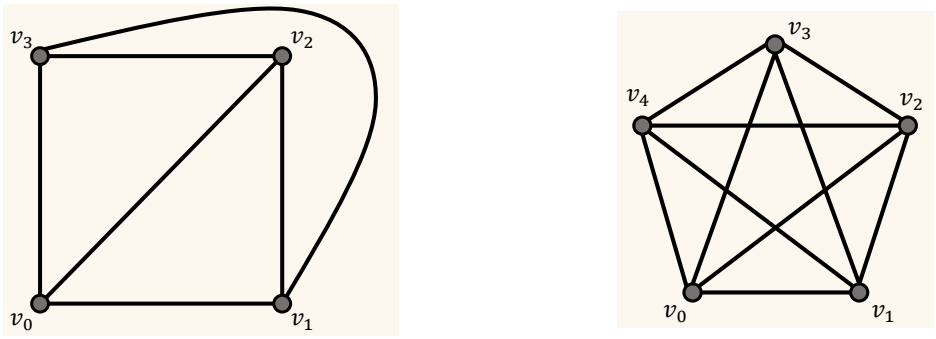
[2] Tamal K Dey, Fengtao Fan, and Yusu Wang. Computing topological persistence for simplicial maps. In *Proceedings of the thirtieth annual symposium on Computational geometry*, pages 345–354, 2014

Keywords : Geometric modeling, Homology, Effective short exact sequences, Topological structures, Chain complexes, Parallelism.

Sommaire

Introduction

Motivations, contexte et objectifs. Avec la montée en puissance des ordinateurs, l'utilisation d'objets virtuels s'est standardisée : synthèse d'image, modélisation, reconstruction, impression 3D etc... Le besoin d'outils d'analyse de ces objets s'est fait grandissant. La topologie algébrique se révèle être une source particulièrement intéressante dans laquelle puiser de tels outils. En s'appuyant sur des notions d'algèbre, elle permet d'associer des propriétés topologiques aux objets, comme par exemple *l'homologie*. Plus précisément, ces propriétés sont des *invariants topologiques*, c'est-à-dire qu'elles sont préservées lorsque l'objet subit des déformations "continues" (translation, rotation, mise à l'échelle etc...). Ces propriétés sont utiles à la conception d'outils d'analyse et peuvent par exemple suffire à montrer que deux objets sont différents (cf. figure 1).



(a) (b)

FIGURE 1 – (a) Un graphe planaire. (b) Un graphe non planaire.

Exemple. La caractéristique d'Euler E d'une subdivision du plan définie par le tracé d'un graphe G connexe, vérifie [3, p.66] :

$$E = v - e + f = 2$$

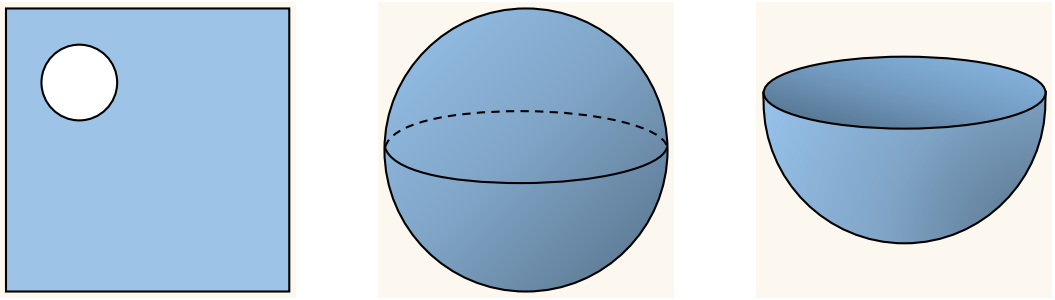
où v désigne le nombre de sommets de G, e son nombre d'arêtes et f son nombre de faces. La notion de face n'a de sens que si G est planaire, et désigne une région du plan délimitée par G. Il découle de cette propriété que si G est un graphe connexe planaire, alors $e \leq 3v - 6$ (en supposant que chaque face a au moins 3 arêtes dans son bord, cf. corollaire 13.4 de [3, p.67]).

Considérons le graphe planaire de la figure 1a, composé de 4 sommets, 6 arêtes et 4 faces (l'extérieur du graphe compte comme une face). Il vérifie $e \leq 3v-6$ car $6 \leq 6$ et $E = 4-6+4 = 2$.

Considérons le graphe de la figure 1b. Il est composé de 5 sommets et 10 arêtes et ne vérifie donc pas $e \leq 3v - 6$ car $10 \leq 9$ est une contradiction. On en déduit que ce graphe n'est pas planaire.

Au travers de cet exemple, nous montrons l'intérêt d'utiliser la caractéristique d'Euler, un invariant topologique, pour analyser un graphe.

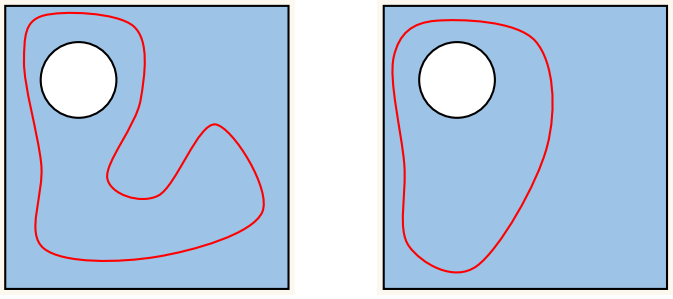
L'invariant auquel nous nous intéressons dans le cadre de cette thèse est l'homologie. Cet invariant est particulièrement puissant de par la quantité d'informations qu'il fournit : composantes connexes, cavités etc... Il est utilisé en modélisation géométrique [4] et dans d'autres domaines comme la médecine [5], la physique [6], l'analyse de données [7, 8, 9] ou encore en apprentissage automatique [10]. Intuitivement, l'homologie d'un objet représente ses "trous". La figure 2 illustre différents objets "troués" et non "troués".



(a) (b) (c)

FIGURE 2 – Les objets (a) et (b) sont troués et l'objet (c) ne l'est pas. (a) Le trou délimite une région du plan, et est *de dimension* 1. (b) La sphère a une cavité, c'est-à-dire *un trou de dimension* 2. (c) L'hémisphère n'a pas de trou.

L'homologie formalise la notion de "trou" : un *trou* est un *cycle* qui n'est pas un *bord*. Intuitivement, on peut s'imaginer un *cycle* comme un chemin qui boucle sur lui-même, c'est-à-dire qu'en partant d'un point de ce chemin et en avançant sans jamais se retourner, on finit par retomber sur le même point. Un *bord* est une délimitation d'une "forme" : deux sommets forment le bord d'une arête, trois arêtes forment le bord d'une face triangulaire etc... L'homologie formalise ces idées au travers de notions d'algèbre, on parle alors de *groupe d'homologie*, calculés à partir de *groupe de cycles* et de *groupe des bords*. Ces notions sont présentées plus en détail dans le Chapitre 1. Notons que plusieurs *cycles* peuvent représenter le même trou, comme l'illustre la figure 3. Un *cycle* est donc **une** représentation d'un trou. Formellement, on appelle ces représentants des *générateurs des groupes d'homologie*.



(a) (b)

FIGURE 3 – En rouge, deux cycles qui représentent le même trou.

Calculer l'homologie d'un objet par ordinateur et de manière efficace se révèle être un challenge auquel de nombreux chercheurs s'intéressent [11, 12, 13, 14, 15]. En modélisation géomé-

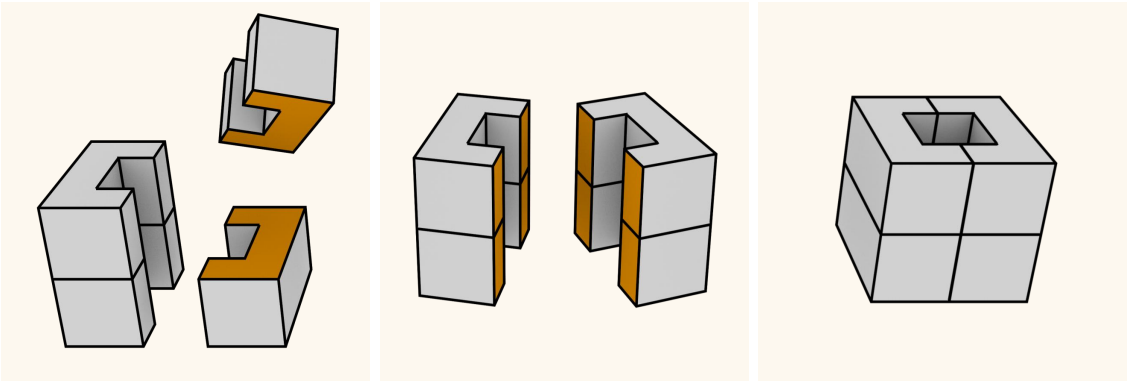
trique, il est fréquent que l'objet manipulé **évolue** par applications d'opérations.

Les travaux que nous présentons s'articulent autour du suivi de l'homologie d'objets géométriques évoluant dans un **processus de construction**, par application d'opérations pouvant potentiellement changer leur homologie. Notre objectif est de calculer, incrémentalement, l'homologie de l'objet à chaque étape de construction, en tenant compte de l'homologie de l'étape qui précède et de l'opération appliquée à l'objet. Les opérations que nous considérons consistent :

— soit à *fusionner* des parties d'objets entre elles ;

— soit à *scinder* des parties d'objets.

En particulier, nous supposons que **les opérations sont locales**, c'est-à-dire que seule une "petite" partie de l'objet est modifiée pour passer d'une étape à la suivante.



(a) Étape 0. (b) Étape 1. (c) Étape 2.

FIGURE 4 – Un processus de construction en 3 étapes. Le passage d'une étape à l'autre se fait en fusionnant les parties d'objet en orange.

Exemple. La figure 4 illustre un processus de construction en 3 étapes. Le passage d'une étape à l'autre se fait par fusion des parties en orange. En particulier, observons que l'objet a successivement 3 composantes connexes, puis 2, puis 1. L'homologie permet de caractériser ces changements structurels.

Les travaux présentés dans cette thèse sont basés sur *l'homologie effective*, un paradigme de calcul d'homologie. Plus particulièrement, les travaux de [1] et [16] sont les points de départ de cette thèse. L'objectif de la thèse est d'explorer la mise en œuvre du théorème des suites exactes courtes effectives (théorème SECE) qui y est présenté et qui permet, à priori, de calculer incrémentalement les variations d'homologie lors de l'application d'opérations comme *l'identification* et *la désidentification*. Les contributions sont :

— *l'analyse du théorème SECE* appliqué à l'identification et à la désidentification. L'un des objectifs de l'analyse est de mettre en évidence les particularités des calculs qu'implique le théorème SECE en vue de l'implémenter efficacement ;

— la *conception de structures de données* adaptées aux particularités d'implémentation du théorème SECE ;

— le *développement d'une bibliothèque logicielle* d'homologie effective, nommée *HomInc*, disponible à l'adresse `https://gitlab.xlim.fr/hominc/HomInc` ;

— l'étude expérimentale d'une implémentation du théorème SECE appliqué à l'identification utilisant *HomInc* ;

— la conception d'un algorithme de *répartition d'identification* dont le but est de permettre *le calcul incrémental* de l'homologie d'un objet éclaté en plusieurs morceaux, distribués sur différentes *unités de calcul*, et sur lequel sont appliquées des identifications ;

— la mise en corrélation de l'homologie effective et de *l'homologie persistante*, un autre paradigme de calcul d'homologie. En particulier, nous mettons en évidence quelques points communs qui rapproche des travaux récents sur l'*homologie persistante de tours* et le théorème SECE.

Remarque. À ce jour, le développement logiciel de l'algorithme de *répartition d'identification* reste à terminer. Nous présentons néanmoins une partie des travaux en cours à ce sujet. Notons également que l'analyse du théorème SECE et l'étude expérimentale ont fait l'objet de communications scientifiques, la plus notable étant [17], publiée au journal *Computer Aided Geometric Design*.

Plan de la thèse. La thèse est composée de 4 chapitres :

— le chapitre 1, dans lequel sont présentées les notions communes aux 3 autres chapitres ;

— le chapitre 2 est composé,

  · d'une partie théorique, dans laquelle nous proposons une analyse du *théorème des suites exactes courtes effectives* (théorème SECE) ;

  · d'une partie pratique, dans laquelle nous présentons les structures de données qui découlent de l'analyse et les résultats expérimentaux obtenus avec une implémentation du théorème SECE basée sur ces structures ;

— le chapitre 3 propose un algorithme de répartition d'une identification appliquée à un objet distribué sur plusieurs unités de calcul. On suppose que l'objet géométrique étudié est éclaté en plusieurs morceaux, chacun assigné à une unité de calcul. L'objectif est de répartir une identification appliquée à l'objet géométrique sur sa distribution de manière à ce qu'il ne soit jamais nécessaire de le reconstruire pour calculer son homologie ;

— le chapitre 4 met en corrélation l'homologie effective et l'homologie persistante en présentant notamment un cadre commun aux deux paradigmes.

Remarque. Sur support numérique, les références sont des liens hypertextes sur lesquels il est possible de cliquer pour naviguer dans le document.

Chapitre 1

Contexte scientifique

Contenu du chapitre

1.1 Notions préliminaires

Cette section présente les notions d'algèbre utilisées par la suite. Par souci de clarté, nous limitons les définitions et propriétés à ce que nous utilisons. En particulier, certains axiomes de la théorie des ensembles ne sont pas rappelés. Nous invitons le lecteur à se référer aux documents cités s'il souhaite approfondir ces notions.

1.1.1 Ensembles, applications, morphismes, graduation

Les notions d'ensembles et d'applications sont supposées connues. Soient F et G deux ensembles et $\phi : F \to G$ une application de F sur G. F est le **domaine** de l'application et G est son **codomaine**. Nous distinguons quelques applications particulières :

— si $F = G$, $\forall x \in F$, $x\phi = x$, alors ϕ est l'**identité** sur F ;

— si $F \subset G$, $\forall x \in F$, $x\phi = x$, alors ϕ est une **inclusion** de F dans G.

Plus généralement, on peut vouloir associer une structure algébrique à une autre. La définition ci-dessous s'inspire de [18] page 53 et 766. Une structure algébrique, telle que nous la manipulons, est un ensemble muni d'applications internes.

Définition 1.1.1. *Soient deux structures algébriques $S_F = (F, (f_i)_{0\leq i})$ et $S_G = (G, (g_i)_{0\leq i})$. Pour tout i, notons d_i l'arité de l'application f_i. Un **morphisme** ϕ de S_F sur S_G est une application $\phi : F \to G$ vérifiant :*

$$(x_1, \ldots, x_{d_i}) f_i \phi = (x_1\phi, \ldots, x_{d_i}\phi) g_i, \qquad \forall i | 0 \leq i, \forall (x_1, \ldots, x_{d_i}) \in F^{d_i}$$

Remarque. Les notions de domaine, codomaine, identité et inclusion vues pour les applications s'étendent aux morphismes.

Définition 1.1.2. *Un **ensemble gradué** E est une union d'ensembles disjoints $(E^p)^{p\geq 0}$.*

Définition 1.1.3. *Soient deux ensembles gradués $F = (F^p)^{p\geq 0}$ et $G = (G^p)^{p\geq 0}$. L'**application** $\phi : F \to G$ **est graduée de degré** $k \in \mathbb{Z}$ si elle vérifie :*

$$\phi : F^p \to G^{p+k}, \qquad \forall p | p \geq 0, p + k \geq 0,$$

Définition 1.1.4. *Soient deux structures algébriques $S_F = (F, (f_i)_{0\leq i\leq n})$ et $S_G = (G, (g_i)_{0\leq i\leq n})$, telles que F et G sont deux ensembles gradués. ϕ est **un morphisme gradué de degré** $k \in \mathbb{Z}$ si :*

— *$\phi : F \to G$ est une application graduée de degré k ;*

— *$(x_1, \ldots, x_{d_i}) f_i \phi = (x_1\phi, \ldots, x_{d_i}\phi) f_{i+k}$, $\forall i | i \geq 0, i + k \geq 0$, $(x_1, \ldots, x_{d_i}) \in F^{d_i}$;*

Remarque. Par la suite, le degré n'est pas précisé lorsqu'il vaut 0. Dans ce cas, on parle plus simplement d'application graduée et de morphisme gradué.

1.1.2 Groupes abéliens

Un groupe abélien est une structure algébrique munie d'une seule application interne. La définition ci-dessous s'inspire de [18] page 4 et 7.

Définition 1.1.5. *Un **groupe abélien** $G = (X, +)$ est un ensemble X muni d'une opération $+$ commutative et associative tel que :*

— *il existe un élément neutre 0_X de X pour l'opération $+$;*

— *pour tout élément de X, il existe un élément inverse de X pour l'opération $+$.*

Exemple. L'ensemble des entiers relatifs $\mathbb{Z}$ muni de l'addition usuelle $+$ est un groupe abélien. L'élément neutre est 0. Pour tout x de $\mathbb{Z}$, l'élément inverse de x est $-x$.

La propriété ci-dessous est issue de l'exemple que donne [18] page 118. Un **module**[1] est une structure algébrique vérifiant certaines propriétés, et composée :

— d'un groupe abélien $G = (X, +)$;

— d'un anneau $(A, +_A, \times_A)$;

— d'une opération $\times : (X, A) \to X$.

Propriété 1.1. À tout groupe abélien $G = (X, +)$ correspond un module composé de lui-même, de l'anneau $(\mathbb{Z}, +_\mathbb{Z}, \times_\mathbb{Z})$, et de l'opération $\times : (X, \mathbb{Z}) \to X$ définie par :

$$x \times z = \underbrace{x + x + \ldots + x + x}_{z \text{ fois}} \qquad \forall x \in X, \forall z \in \mathbb{Z}$$

Remarque. Par la suite, on ne différencie pas un groupe abélien du module qui lui est associé. Dit autrement, nous notons zx l'élément $\underbrace{x + x + \ldots + x + x}_{z \text{ fois}}$ de X.

Définition 1.1.6. *Un **groupe abélien gradué** $G = (X, +)$ est une somme directe de groupes abéliens :*

$$G = \bigoplus_{p \geq 0} G^p$$

où $X = (X^p)^{p \geq 0}$ est un ensemble gradué et $G^p = (X^p, +^p)$.

Par définition, la notion de morphisme s'applique aux groupes abéliens, de même que la notion de morphisme gradué s'applique aux groupes abéliens gradués. Soient deux groupes abéliens gradués $G_X = (X, +_X)$ et $G_Y = (Y, +_Y)$; si ϕ est un morphisme gradué de degré k de G_X sur G_Y, alors :

$$(x_1 +_X x_2)\phi = x_1\phi +_Y x_2\phi, \qquad \forall x_1, x_2 \in X^p | p \geq 0, p + k \geq 0, \text{ et } x_1\phi, x_2\phi \in Y^{p+k}$$

Dans le cas des groupes, le **noyau** de ϕ est l'ensemble des éléments x de X tels que $x\phi = 0_Y$. En particulier, si $\forall x \in X$, $x\phi = 0_Y$, alors ϕ est **un morphisme nul**.

1.2 Structures topologiques

Considérons des objets géométriques subdivisés en cellules de différentes dimensions, c'est-à-dire en sommets, arêtes, faces, volumes, etc... Une structure topologique telle que définie

1. Pour les besoins de cette thèse, il n'est pas nécessaire de comprendre la notion de module en profondeur, qui n'est donc pas définie formellement. Se référer à [18] page 117 pour une définition plus formelle du **module**, et page 83 pour l'anneau.

dans cette section est une représentation de la "structure" d'un objet géométrique subdivisé, c'est-à-dire des cellules de l'objet ainsi que leurs relations d'incidence.

Remarque. Les structures présentées dans cette section sont toutes abstraites, c'est-à-dire qu'elles ne représentent que les informations structurelles et non les informations de forme. Pour cette raison, le terme "abstrait" n'est pas systématiquement précisé.

1.2.1 Structures simpliciales

Les structures simpliciales permettent de représenter la "structure" d'objets subdivisés en simplexes de différentes dimensions. Nous présentons dans cette sous-section :

— les complexes simpliciaux abstraits, très présents dans la littérature. Nous les utilisons en section 1.4 pour faire le lien entre différentes méthodes de calcul de *groupes d'homologie* ;

— les ensembles semi-simpliciaux, une généralisation des complexes simpliciaux abstraits qui sert de base pour la définition d'autres structures plus générales ;

— les ensembles simpliciaux, une généralisation des ensembles semi-simpliciaux. Nous les utilisons pour établir des correspondances avec certaines opérations définies sur les complexes simpliciaux abstraits.

1.2.1-a Complexes simpliciaux abstraits

Les définitions de ce paragraphe s'inspirent de [19].

Définition 1.2.1. *Soit un ensemble fini V, dont les éléments sont appelés sommets. Un **simplexe abstrait** de dimension p, ou p-simplexe abstrait, est un sous-ensemble de V composé de $p+1$ sommets.*

(a) (b) (c) (d)

FIGURE 1.1 – Représentation graphique de simplexes. (a) De dimension 0. (b) De dimension 1. (c) De dimension 2. (d) De dimension 3.

La figure 1.1 illustre des simplexes de différentes dimensions. Soient σ et τ deux simplexes, σ est **une face** de τ et τ est une **coface** de σ si $\sigma \subseteq \tau$.

Définition 1.2.2. *Soit V un ensemble de sommets. Un **complexe simplicial abstrait** $\mathbb{K}$ induit par V est un ensemble de simplexes dont les sommets appartiennent à V, tel que pour tout simplexe σ de $\mathbb{K}$, toutes les faces de σ appartiennent à $\mathbb{K}$.*

La **dimension** d'un complexe simplicial abstrait $\mathbb{K}$ est celle des simplexes de dimension maximale de $\mathbb{K}$.

Exemple. Soit V l'ensemble $\{v_0, v_1, v_2, v_3\}$:

— $\{v_0\}$ est un 0-simplexe ;

— $\{v_0, v_1\}$ est un 1-simplexe ;

— $\{v_0, v_1, v_2\}$ est un 2-simplexe.

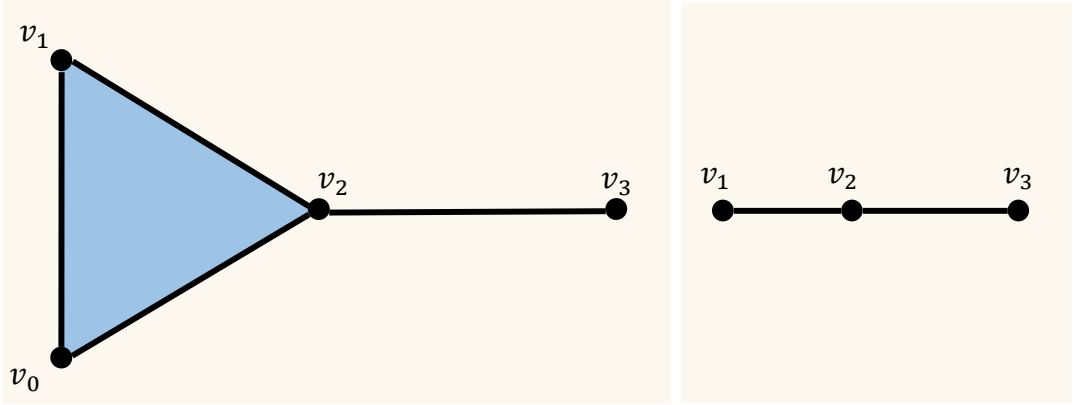
En particulier, $\{v_0\}$ est une face de $\{v_0, v_1\}$ et de $\{v_0, v_1, v_2\}$. Ces derniers sont des cofaces de $\{v_0\}$. L'ensemble $\mathbb{K} = \{\{v_0, v_1\}, \{v_0\}, \{v_1\}\}$ est un complexe simplicial abstrait de dimension 1. L'ensemble $\mathbb{K}' = \{\{v_0, v_1\}, \{v_1\}\}$ n'est pas un complexe simplicial abstrait car $\{v_0\}$ est une face de $\{v_0, v_1\}$, mais elle n'appartient pas à $\mathbb{K}$.

Définition 1.2.3. *Soient $\mathbb{K}$ et $\mathbb{K}'$ deux complexes simpliciaux abstraits induits respectivement par V et V'. L'application $\phi : V \to V'$ est une* ***application simpliciale*** *si elle vérifie :*

$$\forall \sigma = \{v_0, \dots, v_p\} \in \mathbb{K}, \sigma' = \{v_0\phi, \dots, v_p\phi\} \in \mathbb{K}'$$

Par extension, on note aussi ϕ l'application de $\mathbb{K}$ dans $\mathbb{K}'$ qui associe σ à σ'.

Remarque. Notons que comme σ est un p-simplexe de K, $\sigma' = \sigma\phi$ est un k-simplexe de K' tel que $k \leq p$.



(a) (b)

FIGURE 1.2 – (a) Une représentation graphique du complexe simplicial abstrait $\mathbb{K} = \{\{v_0\}, \{v_1\}, \{v_2\}, \{v_3\}, \{v_0, v_1\}, \{v_0, v_2\}, \{v_1, v_2\}, \{v_2, v_3\}, \{v_0, v_1, v_2\}\}$ induit par $V = \{v_0, v_1, v_2, v_3\}$. (b) Une représentation graphique de $\mathbb{K}' = \{\{v_1\}, \{v_2\}, \{v_3\}, \{v_1, v_2\}, \{v_2, v_3\}\}$ induit par $V' = \{v_1, v_2, v_3\}$. (a,b) L'application $\phi : V \to V'$ définie par $v_0\phi = v_1$, $v_1\phi = v_1$, $v_2\phi = v_2$, $v_3\phi = v_3$ est une application simpliciale. L'application $\phi' : V \to V'$ définie par $v_0\phi' = v_3$, $v_1\phi' = v_1$, $v_2\phi' = v_2$ et $v_3\phi' = v_3$ n'est pas une application simpliciale, car $\{v_0, v_1\}\phi' = \{v_3, v_1\}$ n'est pas un simplexe de $\mathbb{K}'$.

1.2.1-b Ensembles semi-simpliciaux

Les définitions de ce paragraphe s'inspirent de [20].

Définition et terminologie.

Définition 1.2.4. *Un* ***ensemble semi-simplicial*** *de dimension n est un couple $(K, d)^n$ tel que :*

— $K = \bigcup_{0 \leq p \leq n} K^p$ *est un ensemble gradué ;*

— $d : K \to K$ *est une application graduée* $(d_i^p)_{0\leq i\leq p}^{1\leq p\leq n}$ *de degré* -1 *vérifiant :*

$$d_i^p d_j^{p-1} = d_j^p d_{i-1}^{p-1} \qquad \forall i, j, p | 1 \leq p \leq n, 0 \leq j < i \leq p$$

Un élément de K^p est un p-**simplexe**. Une application d_i^p est appelée **opérateur de face**. Un p-simplexe σ est **une face** d'un p'-simplexe τ et τ une **coface** de σ si σ peut être obtenu par l'application sur τ d'une composition d'opérateurs de face (à noter que $p < p'$). Le **bord** d'un p-simplexe est l'ensemble de ses faces. **L'étoile** d'un p-simplexe est l'ensemble de ses cofaces. Deux simplexes sont incidents si l'un est une face de l'autre.

Remarque. Par la suite :

— la dimension p d'un simplexe n'est précisée que si cela est utile ;

— l'exposant d'un opérateur de face est parfois omis ;

— un ensemble semi-simplicial de dimension n est parfois noté (K, d).

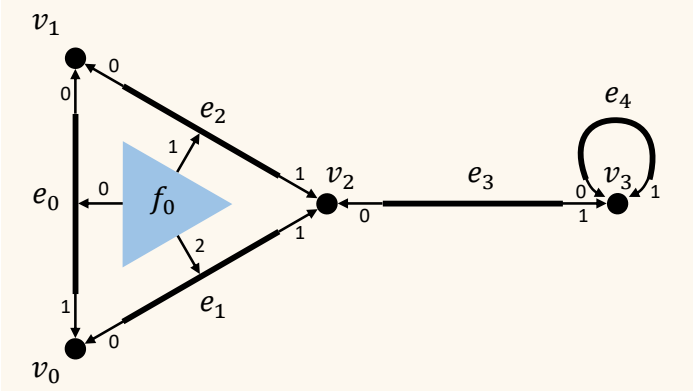


FIGURE 1.3 – Représentation graphique d'un ensemble semi-simplicial $(K, d)^2$ dans laquelle les opérateurs de face $(d_i)_{0\leq i\leq 2}$ sont représentés par des flèches annotées par i.

Exemple. La figure 1.3 illustre un ensemble semi-simplicial $(K, d)^2$. Les simplexes de K sont :

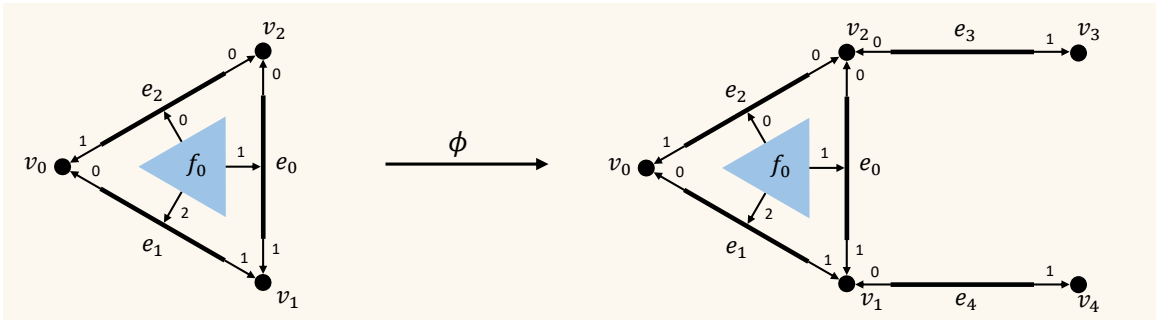
$$K^0 = \{v_0, v_1, v_2, v_3\}$$
$$K^1 = \{e_0, e_1, e_2, e_3, e_4\}$$
$$K^2 = \{f_0\}$$

Sur cet exemple, le 1-simplexe e_2 est une face de f_0 car $f_0 d_1 = e_2$. En particulier, les opérateurs de face vérifient : $f_0 d_1 d_0 = f_0 d_0 d_0 = v_1$, $f_0 d_2 d_0 = f_0 d_0 d_1 = v_0$ et $f_0 d_2 d_1 = f_0 d_1 d_1 = v_2$.

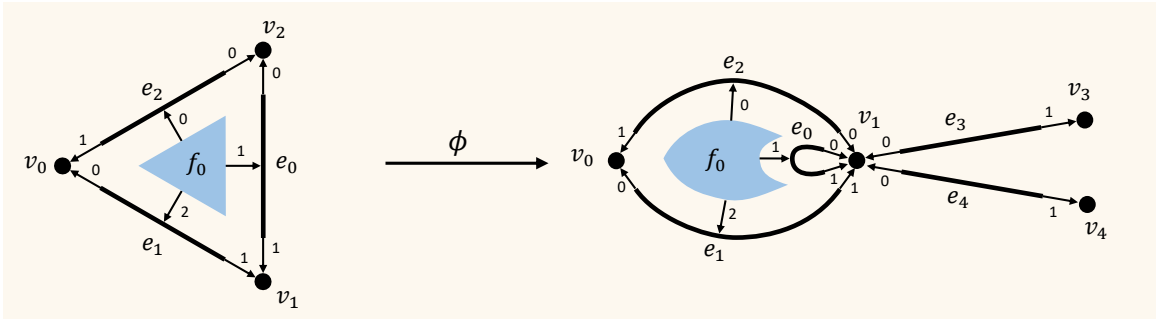
Définition 1.2.5. *Soient* $S = (K, d)^n$ *et* $S' = (K', d')^{n'}$ *deux ensembles semi-simpliciaux, avec* $n \leq n'$. *Une* ***application simpliciale*** *de* S *sur* S' *est un morphisme gradué de* S *sur* S'. *En particulier,* $\phi : K \to K'$ *vérifie :*

$$d_i^p \phi = \phi d_i'^p \qquad \forall i, p | 1 \leq p \leq n, 0 \leq i \leq p$$

La figure 1.4 illustre deux exemples d'applications simpliciales.



(a)



(b)

FIGURE 1.4 – (a) Une application simpliciale $\phi : K \to K'$ définie par $v_0\phi = v_0$, $v_1\phi = v_1$, $v_2\phi = v_2$, $e_0\phi = e_0$, $e_1\phi = e_1$, $e_2\phi = e_2$, $f_0\phi = f_0$. ϕ est une inclusion. (b) Une application simpliciale $\phi : K \to K'$ défini par $v_0\phi = v_0$, $v_1\phi = v_2\phi = v_1$, $e_0\phi = e_0$, $e_1\phi = e_1$, $e_2\phi = e_2$, $f_0\phi = f_0$.

Construction d'un ensemble semi-simplicial à partir d'un complexe simplicial abstrait. Soit $V = \{v_0, \ldots, v_q\}$, un ensemble de sommets sur lequel un ordre est défini, représenté ici par les indices des sommets. On peut associer à tout complexe simplicial abstrait $\mathbb{K}$ induit par V un ensemble semi-simplicial (K, d) :

— à tout simplexe de $\mathbb{K}$ est associé dans K un simplexe correspondant à la suite de ses sommets ordonnés par ordre croissant ;

— pour tout p-simplexe σ de K, si σ est associé à la suite croissante $(v_{x_0}, \ldots, v_{x_i}, \ldots, v_{x_p})$, alors σd_i est le $(p-1)$-simplexe associé à la suite $(v_{x_0}, \ldots, v_{x_{i-1}}, v_{x_{i+1}}, \ldots, v_{x_p})$.

Exemple. Soit un complexe simplicial abstrait défini par :

$$\mathbb{K} = \{\{v_0\}, \{v_1\}, \{v_2\}, \{v_3\}, \{v_0, v_1\}, \{v_1, v_2\}, \{v_0, v_2\}, \{v_2, v_3\}, \{v_0, v_1, v_2\}\}$$

On peut lui associer l'ensemble semi-simplicial (K, d) défini par[2] :

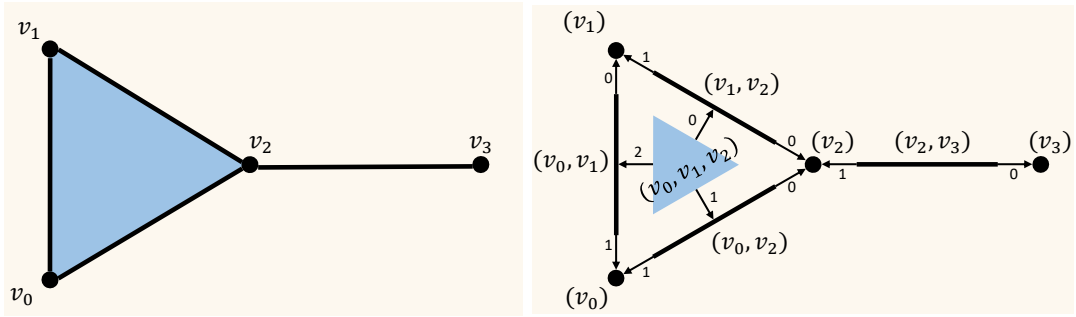
— $K = \{(v_0), (v_1), (v_2), (v_3), (v_0, v_1), (v_1, v_2), (v_0, v_2), (v_2, v_3), (v_0, v_1, v_2)\}$

— les opérateurs de face sont définis par :

· $(v_0, v_1, v_2)d_0 = (v_1, v_2)$, $(v_0, v_1, v_2)d_1 = (v_0, v_2)$, $(v_0, v_1, v_2)d_2 = (v_0, v_1)$;

· $(v_0, v_1)d_0 = (v_1)$, $(v_0, v_1)d_1 = (v_0)$, $(v_1, v_2)d_0 = (v_2)$, $(v_1, v_2)d_1 = (v_1)$, $(v_0, v_2)d_0 = (v_2)$, $(v_0, v_2)d_1 = (v_0)$, $(v_2, v_3)d_0 = (v_3)$, $(v_2, v_3)d_1 = (v_2)$.

$\mathbb{K}$ et (K, d) sont illustrés sur la figure 1.5.



(a) (b)

FIGURE 1.5 – (a) Une représentation graphique de $\mathbb{K} = \{\{v_0\}, \{v_1\}, \{v_2\}, \{v_3\}, \{v_0, v_1\}, \{v_1, v_2\}, \{v_0, v_2\}, \{v_2, v_3\}, \{v_0, v_1, v_2\}\}$ défini sur $V = \{v_0, v_1, v_2, v_3\}$ (l'ordre sur V correspond aux indices des sommets). (b) Une représentation graphique de l'ensemble semi-simplicial associé à $\mathbb{K}$.

Remarque. Cette méthode de construction est une adaptation de l'exemple 1.4 de [21, p.2]. Notons qu'à tout complexe simplicial abstrait correspond donc un ou plusieurs ensembles semi-simpliciaux. L'inverse n'est pas vrai. Dans l'exemple de la figure 1.4b, le 1-simplexe e_0 est incident deux fois au 0-simplexe v_1 : il n'est pas possible de lui associer directement un complexe simplicial abstrait. Plus généralement, dès lors qu'un ensemble semi-simplicial contient au moins un simplexe incident plusieurs fois à un autre simplexe (on parle de multi-incidence), il n'est pas possible de leur associer un complexe simplicial abstrait. C'est aussi le cas quand il existe au moins deux simplexes distincts ayant le même bord.

Opérations de construction. Tout ensemble semi-simplicial peut être construit à l'aide des opérations de cône et d'identification présentées ci-dessous [22].

Cône. Étant donné un ensemble semi-simplicial $(K, d)^n$ et un 0-simplexe v, le cône de $(K, d)^n$ sur v est un ensemble semi-simplicial $(K', d')^{n+1}$ composé des simplexes de K, du 0-simplexe v, et d'autant de "nouveaux" simplexes qu'en compte K. Intuitivement, ces derniers peuvent être considérés comme les simplexes de K, "montés" d'une dimension, qui "relient" chaque simplexe de K au 0-simplexe v.

Définition 1.2.6. *Le **cône** d'un ensemble semi-simplicial $(K, d)^n$ et d'un 0-simplexe v est l'ensemble semi-simplicial $(K', d')^{n+1}$ défini par :*

— $K' = K \cup \widehat{K} \cup \{v\}$, *où* $\widehat{K^{p+1}}$ *est en bijection*[3] *avec* $K^p \qquad \forall p | 0 \leq p \leq n.$

— d' *est définie par :*

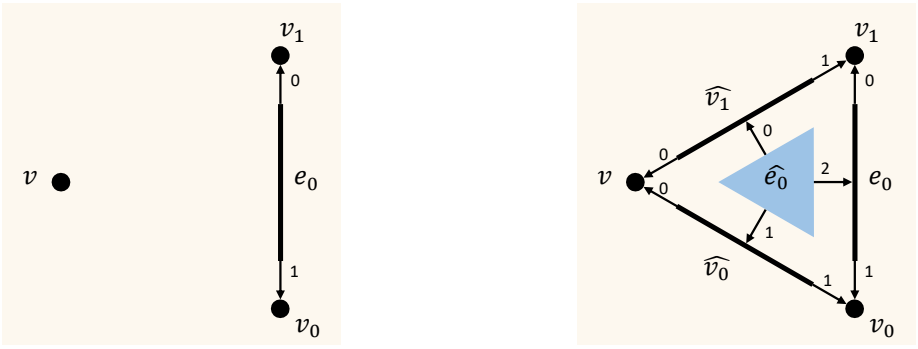
· $\sigma d_i'^p = \sigma d_i^p, \quad \forall \sigma \in K, \forall i, p | 0 \leq i \leq p,\ 1 \leq p \leq n$;

· $\forall \widehat{\sigma} \in \widehat{K^p}$:

▷ *si* $p = 1$, $\widehat{\sigma} d'_0 = v$ *et* $\widehat{\sigma} d'_1 = \sigma$;

▷ *sinon,* $\widehat{\sigma} d'_p = \sigma$ *et* $\widehat{\sigma} d'_i = \widehat{\sigma d_i} \quad \forall i | 0 \leq i \leq p-1.$

2. Chaque simplexe est nommé par la suite ordonnée de ses sommets incidents.
3. La bijection est dénotée par l'accent circonflexe.



(a) (b)

FIGURE 1.6 – Un cône. (a) L'ensemble semi-simplicial $(K,d)^1$ et le 0-simplexe v. (b) Le cône $(K',d')^2$ de $(K,d)^1$ sur v.

Exemple. La figure 1.6 illustre le cône d'un ensemble semi-simplicial $(K,d)^1$ et d'un 0-simplexe v, avec :

$$K^0 = \{v_0, v_1\} \qquad \widehat{K^1} = \{\widehat{v_0}, \widehat{v_1}\}$$
$$K^1 = \{e_0\} \qquad \widehat{K^2} = \{\widehat{e_0}\}$$

En particulier, d' vérifie :

- $\widehat{e_0}d'_0 = \widehat{e_0 d_0} = \widehat{v_1}$
- $\widehat{e_0}d'_1 = \widehat{e_0 d_1} = \widehat{v_0}$
- $\widehat{e_0}d'_2 = e_0$

Identification. Soit un ensemble semi-simplicial $(K,d)^n$. Intuitivement, l'opération d'identification consiste à "fusionner" des simplexes de K ayant le même bord. L'identification de deux p-simplexes σ_1 et σ_2 consiste à :

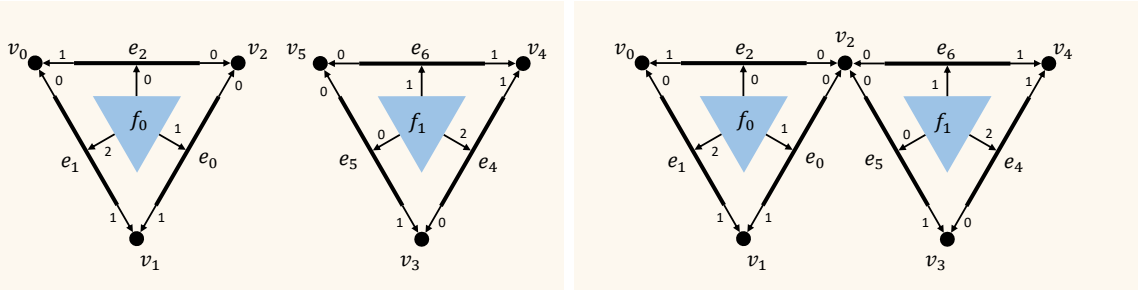
- supprimer σ_2 de K^p ;
- modifier le bord de tout simplexe $\tau \in K^{p+1}$ tel que σ_2 est une face de τ, en remplaçant σ_2 par σ_1.

Définition 1.2.7. *Soit $(K,d)^n$ un ensemble semi-simplicial, et σ_1, σ_2, deux p-simplexes de K^p vérifiant : si $p \geq 1$, alors $\sigma_1 d_i = \sigma_2 d_i$ pour tout i compris entre 0 et p. Le résultat de* ***l'identification de*** *σ_1* ***et*** *σ_2 est l'ensemble semi-simplicial $(K',d')^n$ défini par :*

- $K' = K \setminus \{\sigma_2\}$;
- d' *: pour tout indice i valide et $\sigma \in K'$,*
  - $\sigma d'_i = \sigma_1$ *si* $\sigma d_i = \sigma_2$;
  - $\sigma d'_i = \sigma d_i$ *sinon.*

Nous dirons que σ_1 est le simplexe **survivant** de l'identification, et que le simplexe σ_2 est le simplexe **non-survivant**.

Exemple. La figure 1.7 illustre l'application d'une identification sur un ensemble semi-simplicial $(K,d)^2$. Le 0-simplexe v_5 est non-survivant. En particulier :



(a) (b)

FIGURE 1.7 – Identification. (a) L'ensemble semi-simplicial initial $(K, d)^2$. (b) L'ensemble semi-simplicial $(K', d')^2$ résultant de l'identification de v_2 avec v_5.

— $e_5 d_0 = v_5$ et $e_5 d'_0 = v_2$;
— $e_6 d_0 = v_5$ et $e_6 d'_0 = v_2$.

Plus généralement, on peut vouloir identifier :

— un nombre quelconque de p-simplexes partageant le même bord ;
— deux p-simplexes σ_1 et σ_2 de bords différents. Dans ce cas il est nécessaire d'identifier leurs bords de manière cohérente : s'il existe i, $0 \leq i \leq p$ tel que $\sigma_1 d_i \neq \sigma_2 d_i$ alors $\sigma_1 d_i$ et $\sigma_2 d_i$ sont identifiés. Récursivement, si les bords de $\sigma_1 d_i$ et $\sigma_2 d_i$ sont différents, il faut les identifier de manière cohérente etc. ...
— en toute généralité, un nombre quelconque de simplexes de bords différents. Dans ce cas, on peut caractériser une identification par une famille de partitions $I = (I^p)^{0 \leq p \leq n}$ et une application graduée $\zeta : K \to K$ associant à tout p-simplexe un p-simplexe survivant. I^p est une partition de K^p, où, pour tout p :
  · un élément de I^p est un ensemble de p-simplexes tous identifiés entre eux. Si un élément ne comporte qu'un p-simplexe, celui-ci n'est identifié avec aucun autre p-simplexe ;
  · si σ_1 et σ_2 appartiennent au même élément de I^p, avec $p \geq 1$, alors, pour tout i tel que $0 \leq i \leq p$, $\sigma_1 d_i$ et $\sigma_2 d_i$ appartiennent au même élément de I^{p-1} ;
  · un élément de I^p est noté $\{\sigma_0, \ldots, \sigma_q\}$. Pour tout i, j compris entre 0 et q, $\sigma_i \zeta = \sigma_j \zeta$

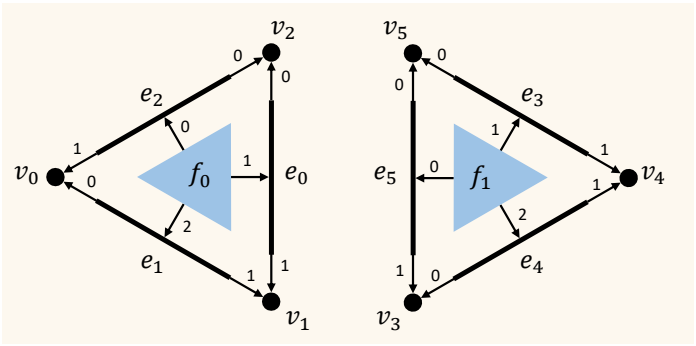
Convention. Pour un élément $\{\sigma_0, \ldots, \sigma_q\}$ de I^p, σ_0 est le simplexe survivant.

Exemple. La figure 1.8 illustre l'application d'une identification sur un ensemble semi-simplicial $(K, d)^2$, caractérisée par $I = (I^p)^{0 \leq p \leq 2}$ telle que :

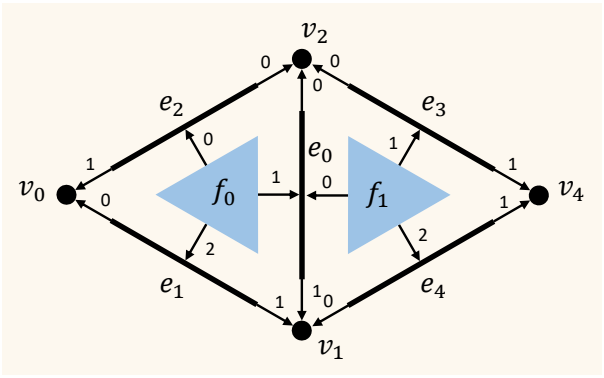
$$I^0 = \{\{v_0\}, \{v_1, v_3\}, \{v_2, v_5\}, \{v_4\}\}$$
$$I^1 = \{\{e_0, e_5\}, \{e_1\}, \{e_2\}, \{e_3\}, \{e_4\}\}$$
$$I^2 = \{\{f_0\}, \{f_1\}\}$$

En particulier, $v_0\zeta = v_0$, $v_1\zeta = v_3\zeta = v_1$, $v_2\zeta = v_5\zeta = v_2$, $e_0\zeta = e_5\zeta = e_0$. L'ensemble gradué K' est défini par :

$$K'^0 = \{v_0, v_1, v_2, v_4\} = K^0 \setminus \{v_3, v_5\}$$
$$K'^1 = \{e_0, e_1, e_2, e_3, e_4\} = K^1 \setminus \{e_5\}$$
$$K'^2 = \{f_0, f_1\} = K^2$$



(a)



(b)

FIGURE 1.8 – Identification. (a) L'ensemble semi-simplicial initial $(K, d)^2$. (b) L'ensemble semi-simplicial $(K', d')^2$ résultant de l'identification de v_1 avec v_3, v_2 avec v_5 et e_0 avec e_5.

Notons que :

— $e_4 d_0 = v_3$ et $e_4 d'_0 = v_1$;
— $e_3 d_0 = v_5$ et $e_3 d'_0 = v_2$;
— $f_1 d_0 = e_5$ et $f_1 d'_0 = e_0$.

Définition 1.2.8. *Soit un ensemble semi-simplicial $(K, d)^n$, une famille de partitions $I = (I^p)^{0 \leq p \leq n}$ et une application graduée $\zeta : K \to K$ tels que : pour tout p valide,*

— *I^p est une partition de K^p ;*
— *$\forall c = \{\sigma_0, \ldots, \sigma_q\} \in I^p$, $\forall \sigma, \sigma' \in c$:*
  · *$\sigma\zeta = \sigma'\zeta$;*
  · *$\sigma d_i, \sigma' d_i$ appartiennent au même élément de I^{p-1}, $\forall p \geq 1, \forall i | 0 \leq i \leq p$.*

L'ensemble semi-simplicial $(K', d')^n$, résultant de ***l'identification caractérisée par*** *I* ***et*** *ζ, est défini par :*

— $\forall \sigma \in K^p, \sigma\zeta \in K'^p$ *et* $|K'^p| = |I^p| \qquad \forall p | 0 \leq p \leq n$;
— $\sigma d'_i = \sigma d_i \zeta, \quad \forall \sigma \in K'^p, \forall i, p | p \geq 1, 0 \leq i \leq p,$.

Désidentification. La désidentification est l'opération inverse de l'identification. Intuitivement, elle consiste à "éclater" les simplexes d'un ensemble semi-simplicial et à modifier les opérateurs de face des simplexes de l'étoile des simplexes éclatés. La figure 1.9 illustre l'application de différentes désidentifications sur un ensemble semi-simplicial (K, d).

Remarque. En toute généralité, les simplexes éclatés ne suffisent pas à eux seuls à définir une désidentification. Pour chaque simplexe éclaté, il faut nécessairement expliciter le nombre de simplexes résultant de son éclatement et la modification des opérateurs de face des simplexes de son étoile, comme l'illustre la figure 1.9 Nous n'utilisons pas ces paramètres dans la suite de cette thèse. Pour cette raison, nous avons choisi de simplifier ce paragraphe et de ne pas donner de définition formelle de la désidentification. À notre connaissance, il n'existe pas de définition de la désidentification qui puisse être référencée. Dans notre contexte, une opération de désidentification est entièrement définie par l'identification inverse qu'elle induit.

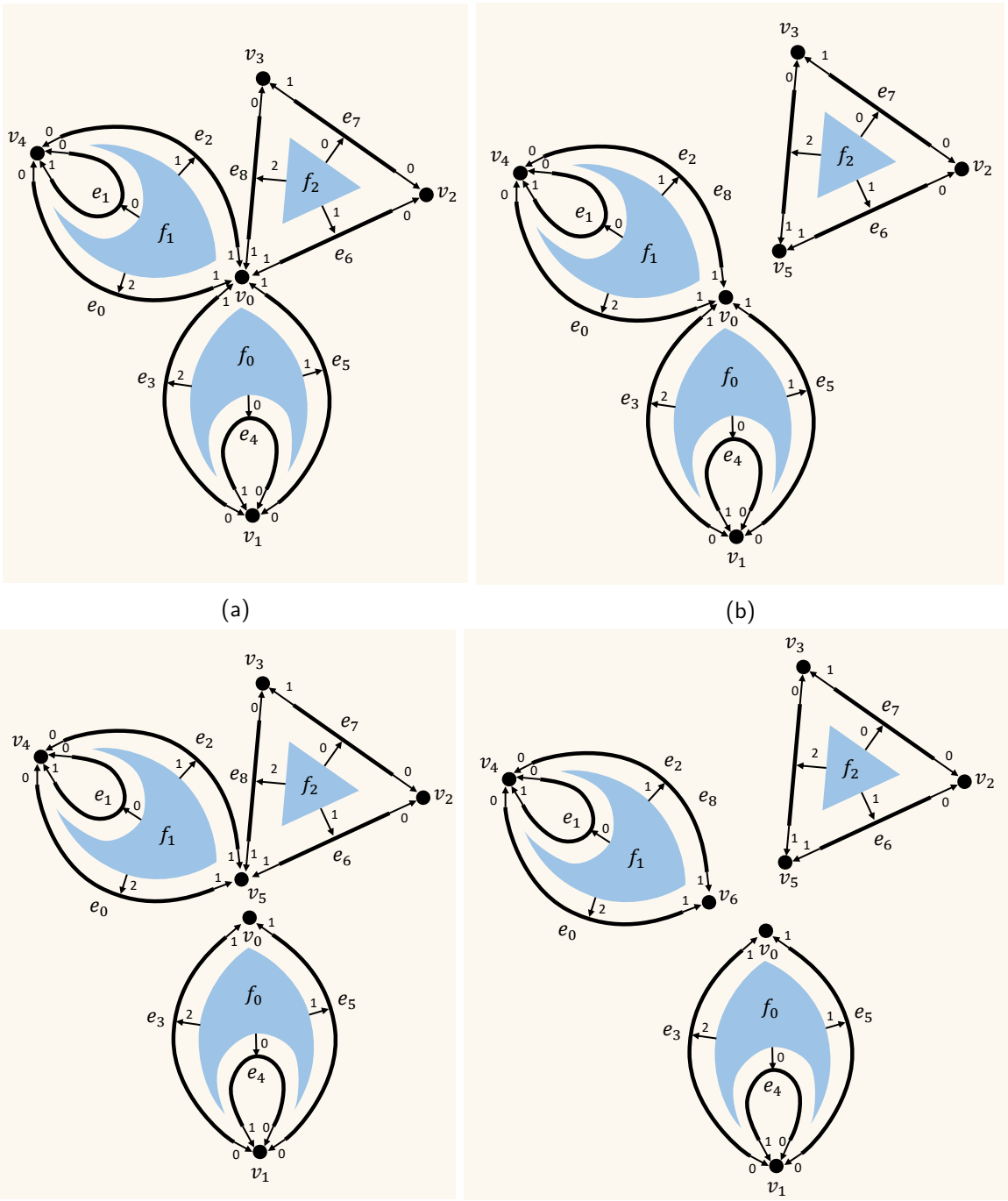


FIGURE 1.9 – Différentes désidentifications appliquées à un ensemble semi-simplicial (K, d). (a) (K, d). (b,c) L'éclatement de v_0 en deux 0-simplexes v_0 et v_5. (d) L'éclatement de v_0 en trois 0-simplexes v_0, v_5 et v_6.

1.2.1-c Ensembles simpliciaux.

Les ensembles simpliciaux généralisent les ensembles semi-simpliciaux en les complétant par des *opérateurs de dégénérescence*. Les définitions de ce paragraphe s'inspirent de [21, 23].

Définition et terminologie.

Définition 1.2.9. *Un **ensemble simplicial** est un triplet (K, d, s) tel que :*

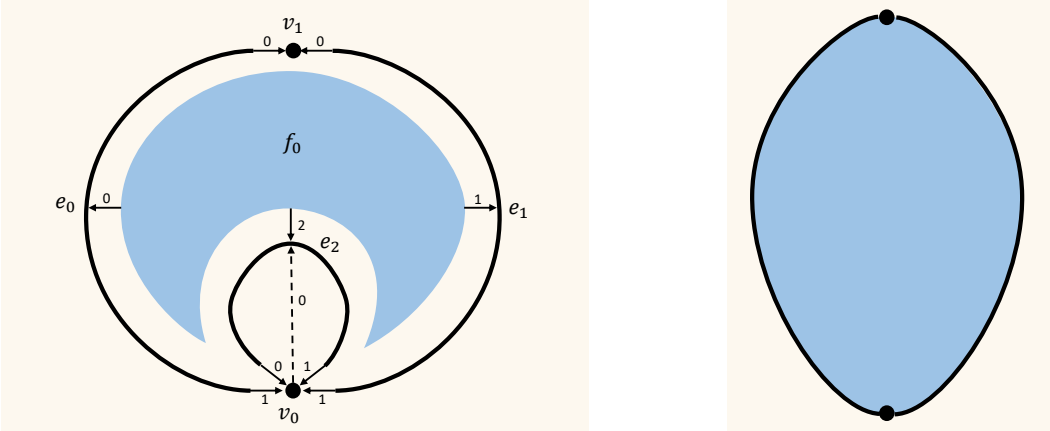
— *(K, d) est un ensemble semi-simplicial ;*

— *$s : K \to K$ est une application graduée $(s_i^p)_{0\leq i\leq p}^{0\leq p}$ de degré 1 vérifiant : pour tous indices i, j valides,*

- $s_i s_j = s_j s_{i+1}$ *si* $i \geq j$;
- $s_i d_j = d_j s_{i-1}$ *si* $i > j$;
- $s_i d_i = s_i d_{i+1} = id_K$;
- $s_i d_j = d_{j-1} s_i$ *si* $i + 1 < j$;

L'application s_i est appelée **opérateur de dégénérescence**. Un simplexe $\tau \in K^{p+1}$ est dit **dégénéré** s'il existe $\sigma \in K^p$ et i compris entre 0 et p tels que $\sigma s_i^p = \tau$.

Remarque. Il existe une infinité de simplexes dégénérés dans un ensemble simplicial non vide.



(a) (b)

FIGURE 1.10 – (a) Un ensemble simplicial (K, d, s) où ne sont représentés que les simplexes dégénérés qui sont des faces de simplexes non dégénérés (seule l'arête e_2 est concernée). Les opérateurs de face d_i sont représentés par des flèches pleines annotées par i. Les opérateurs de dégénérescence s_i sont représentés par des flèches en pointillé annotées par i. (b) Un objet géométrique pouvant être associé à (K, d, s).

Exemple. La figure 1.10a illustre un ensemble simplicial (K, d, s) composé des p-simplexes non dégénérés suivants :

$$K^0 : \{v_0, v_1\}$$
$$K^1 : \{e_0, e_1\}$$
$$K^1 : \{f_0\}$$

En particulier, e_2 est un simplexe dégénéré, car $v_0 s_0 = e_2$.

Définition 1.2.10. *Soient $S = (K, d, s)$ et $S' = (K', d', s')$ deux ensembles simpliciaux. Une **application simpliciale** ϕ de S sur S' est un morphisme gradué de S sur S'. En particulier, $\phi : K \to K'$ vérifie : pour tout indice i valide,*

$$d_i \phi = \phi d'_i$$
$$s_i \phi = \phi s'_i$$

Construction d'un ensemble simplicial à partir d'un complexe simplicial abstrait. Nous avons vu en section 1.2.1-b comment associer un ensemble semi-simplicial $S = (K, d)$ à un complexe simplicial abstrait $\mathbb{K}$ défini sur un ensemble de sommets ordonnés V. L'ensemble simplicial $S' = (K, d, s)$ associé à $\mathbb{K}$ est défini à partir de S en ajoutant les simplexes dégénérés comme suit :

- si σ est un simplexe associé à la suite croissante $(v_{x_0}, \ldots, v_{x_i}, \ldots, v_{x_p})$, alors un simplexe dégénéré σ' est associé à la suite $(v_{x_0}, \ldots, v_{x_i}, v_{x_i}, \ldots, v_{x_p})$, pour tout i compris entre 0 et p ;
- pour tout p-simplexe $\sigma = (v_{x_0}, \ldots, v_{x_p})$:
  - σd_i^p est le $(p-1)$-simplexe associé à la suite $(v_{x_0}, \ldots, v_{x_p})$ privée de v_i ;
  - σs_i^p est le $(p+1)$-simplexe associé à la suite $(v_{x_0}, \ldots, v_{x_i}, v_{x_i}, \ldots, v_{x_p})$;

Remarque. Cette méthode de construction est une adaptation de l'exemple 1.4 de [21, p.2].

Exemple. Le complexe simplicial abstrait illustré sur la figure 1.11 est défini par :

$$\mathbb{K} = \{\{v_0\}, \{v_1\}, \{v_0, v_1\}\}$$

On lui associe l'ensemble semi-simplicial (K, d, s) composé des simplexes non dégénérés (v_0), (v_1), (v_0, v_1). En particulier :

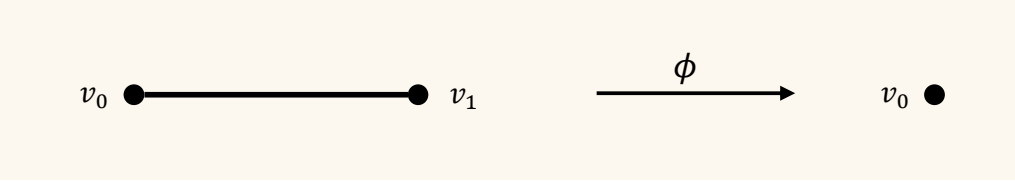
- $(v_0, v_1) d_0 = (v_1)$;
- $(v_0, v_1) s_0 = (v_0, v_0, v_1)$.

L'application simpliciale sur $\mathbb{K}$ est définie par :

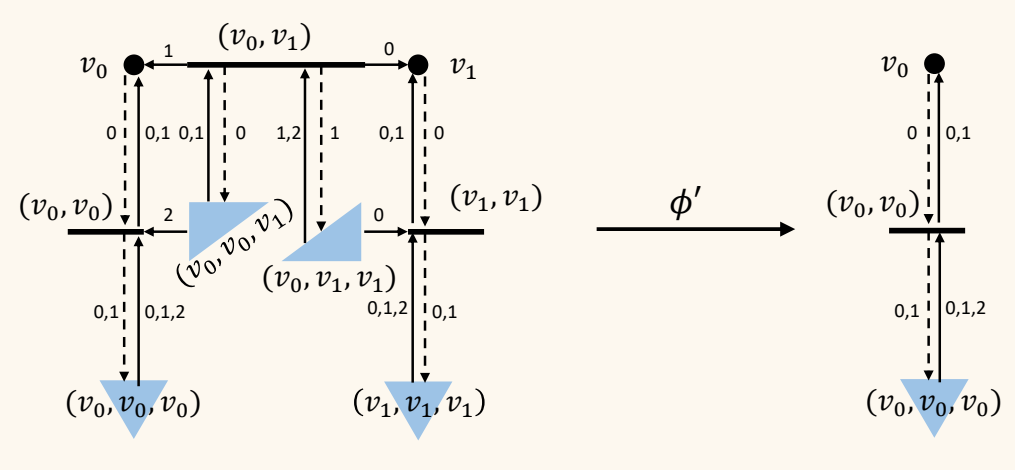
- $v_0 \phi = v_0$;
- $v_1 \phi = v_0$.

En conséquence, ϕ' est principalement définie sur K par :

- $v_0 \phi' = v_0$;
- $v_1 \phi' = v_1$;
- $(v_0, v_1) \phi' = (v_0, v_0)$.



(a)



(b)

FIGURE 1.11 – Une application simpliciale ϕ sur un complexe simplicial abstrait $\mathbb{K}$ et sa correspondance ϕ' sur un ensemble simplicial (K, d, s). Seuls sont représentés les simplexes dégénérés de dimension inférieure ou égale à 2. (a) ϕ (b) ϕ'

1.2.2 Structures cubiques et simploïdales

Un cube géométrique est le produit cartésien de simplexes géométriques de dimension 1. Un simploïde géométrique est le produit cartésien de simplexes de dimensions quelconques. Les structures cubiques (respectivement simploïdales) permettent de représenter la "structure" d'objets subdivisés en cubes (respectivement simploïdes) de différentes dimensions. Nous présentons dans cette sous-section :

— les ensembles semi-cubiques, utilisés en particulier dans le Chapitre 2 ;

— les ensembles semi-simploïdaux, une généralisation des ensembles semi-cubiques. Ils sont présentés à titre indicatif.

Les définitions de cette sous-section s'inspirent de [20].

1.2.2-a Ensembles semi-cubiques

Définition et terminologie.

Définition 1.2.11. *Un **ensemble semi-cubique** de dimension n est un couple $(K, d)^n$ tel que :*

— $K = \bigcup_{0 \leq p \leq n} K^p$ *est un ensemble gradué ;*

— $d : K \to K$ *est une application graduée* $(d^p_{i,j})^{1 \leq p \leq n}_{0 \leq j \leq 1 \leq i \leq p}$ *de degré -1 vérifiant :*

$$d^p_{i,j} d^{p-1}_{x,y} = d^p_{x,y} d^{p-1}_{i-1,j} \qquad \forall i, j, x, y, p | 0 \leq j, y \leq 1 \leq x < i \leq p, 1 \leq p \leq n$$

FIGURE 1.12 – Représentation graphique de cubes de dimension 0 (a), 1 (b), 2(c), 3(d).

Un élément de K^p est un p-**cube**. La figure 1.12 illustre des cubes de différentes dimensions. Une application $d^p_{i,j}$ est appelée **opérateur de face**. Un p-cube σ est une **face** d'un p'-cube τ et τ une **coface** de σ si σ peut être obtenu par l'application sur τ d'une composition d'opérateurs de face (à noter que $p < p'$). Le **bord** d'un p-cube est l'ensemble composé de ses faces. **L'étoile** d'un p-cube est l'ensemble composé de ses cofaces. Deux cubes sont incidents si l'un apparaît dans le bord de l'autre. À noter qu'un cube peut être incident plusieurs fois à un autre cube (cf figure 1.13).

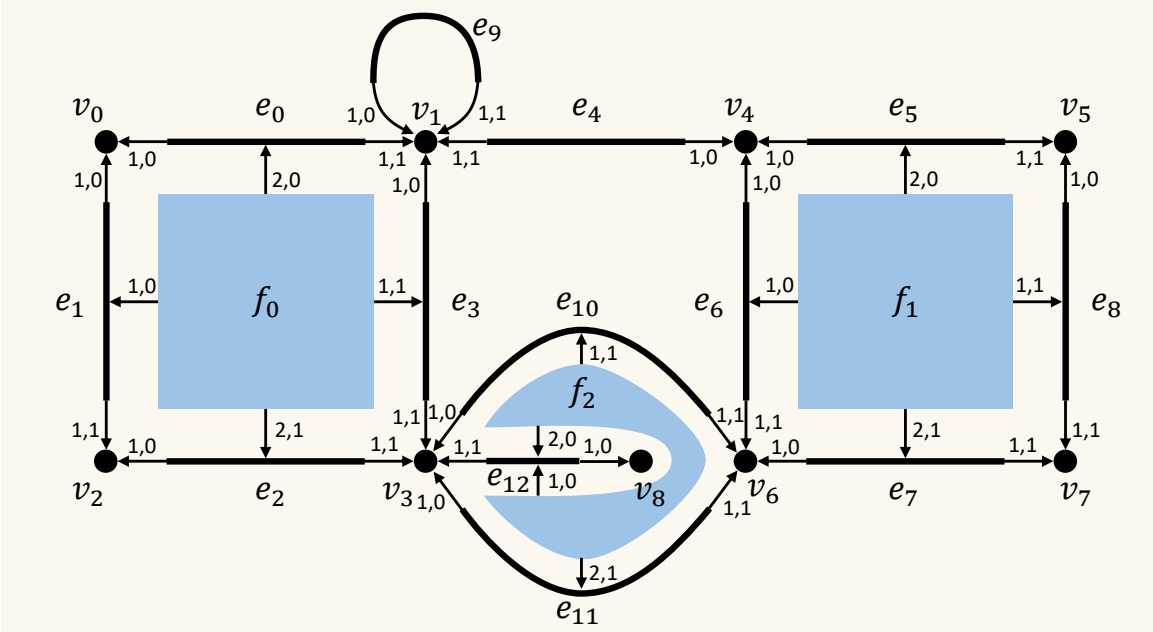


FIGURE 1.13 – Représentation graphique d'un ensemble semi-cubique $(K, d)^2$ dans laquelle les opérateurs de face $(d_{i,j})_{0\leq j\leq 1\leq i\leq 2}$ sont représentés par des flèches annotées par i et j.

Exemple. La figure 1.13 illustre un ensemble semi-cubique $(K, d)^2$. Les cubes de K sont :

$$K^0 = \{v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$$
$$K^1 = \{e_0, e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}\}$$
$$K^2 = \{f_0, f_1, f_2\}$$

En particulier, les opérateurs de face vérifient $f_0 d_{2,0} d_{1,0} = f_0 d_{1,0} d_{1,0} = v_0$ et $f_0 d_{2,0} d_{1,1} = f_0 d_{1,1} d_{1,0} = v_1$. Notons que f_2 est incidente deux fois à e_{12}.

Opérations de construction. Tout ensemble semi-cubique peut être construit en utilisant les opérations de produit cartésien, d'identification et de désidentification.

Produit cartésien.

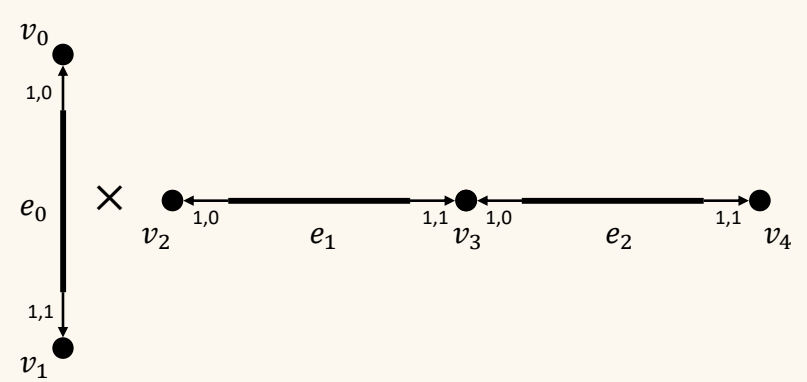
Définition 1.2.12. *Le **produit cartésien** de deux ensembles semi-cubiques $(K,d)^n$ et $(K',d')^{n'}$ est l'ensemble semi-cubique $(K'',d'')^{n+n'}$ défini par :*

— $K'' = K \times K'$ *tel que* $K^p \times K'^l \subset K''^{p+l} \qquad \forall p,l | 0 \leq p \leq n, 0 \leq l \leq n'$;

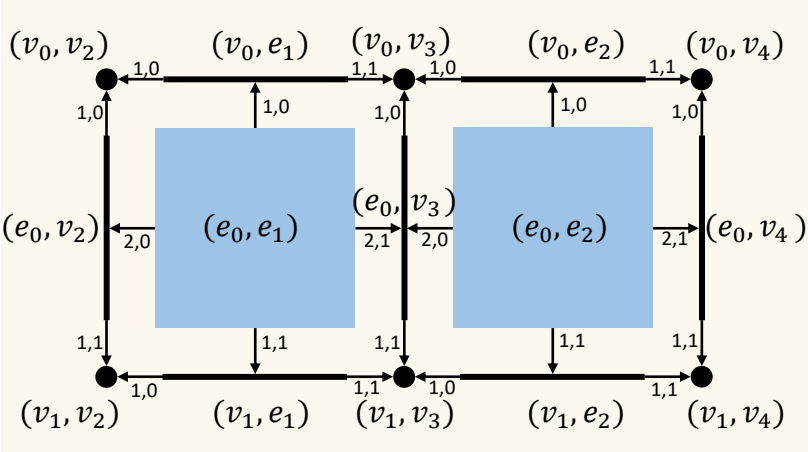
— d'' *est définie par : pour tout* $\sigma \in K^p, \sigma' \in K'^{p'}$,

· $(\sigma,\sigma')d''_{i,j} = (\sigma d_{i,j}, \sigma')$ *si* $i \leq p$;

· $(\sigma,\sigma')d''_{i,j} = (\sigma, \sigma' d'_{i-p,j})$ *sinon.*



(a)



(b)

FIGURE 1.14 – Produit cartésien. (a) Les ensembles semi-cubiques initiaux $(K,d)^1$ et $(K',d')^1$. (b) L'ensemble semi-cubique $(K'',d'')^2$ résultant.

Exemple. La figure 1.14 illustre le produit cartésien de deux ensembles semi-cubiques de dimension 1. Les cubes de K'' sont :

$$K''^0 = \{(v_0,v_2),(v_0,v_3),(v_0,v_4),(v_1,v_2),(v_1,v_3),(v_1,v_4)\}$$
$$K''^1 = \{(e_0,v_2),(e_0,v_3),(e_0,v_4),(v_0,e_1),(v_0,e_2),(v_1,e_1),(v_1,e_2)\}$$
$$K''^2 = \{(e_0,e_1),(e_0,e_2)\}$$

En particulier, notons que :

— $(e_0, e_1)d''_{1,0} = (v_0, e_1)$;
— $(e_0, e_1)d''_{1,1} = (v_1, e_1)$;
— $(e_0, e_1)d''_{2,0} = (e_0, v_2)$;
— et $(e_0, e_1)d''_{2,1} = (e_0, v_3)$.

Identification.

Définition 1.2.13. *Soit $(K, d)^n$ un ensemble semi-cubique, une famille de partitions $I = (I^p)^{0 \leq p \leq n}$ et une application graduée $\zeta : K \to K$ tels que : pour tout p valide,*

— *I^p est une partition de K^p ;*
— *$\forall c = \{\sigma_0, \ldots, \sigma_q\} \in I^p$, $\forall \sigma, \sigma' \in c$,*
  · *$\sigma\zeta = \sigma'\zeta$;*
  · *$\sigma d_{i,j}, \sigma' d_{i,j}$ appartiennent au même élément de I^{p-1}, $\forall p \geq 1$, $\forall i, j | 0 \leq j \leq 1 \leq i \leq p$.*

*L'ensemble semi-cubique $(K', d')^n$ résultant de **l'identification caractérisée par** I **et** ζ est défini par :*

— *$\forall \sigma \in K^p, \sigma\zeta \in K'^p$ et $|K'^p| = |I^p|$;*
— *$\sigma d'_{i,j} = \sigma d_{i,j}\zeta, \quad \forall \sigma \in K'^p$, $\forall i, j, p | p \geq 1, 0 \leq j \leq 1 \leq i \leq p$.*

Exemple. La figure 1.15 illustre l'application d'une identification sur un ensemble semi-cubique $(K, d)^2$ caractérisée par $I = (I^p)^{0 \leq p \leq 2}$ telle que :

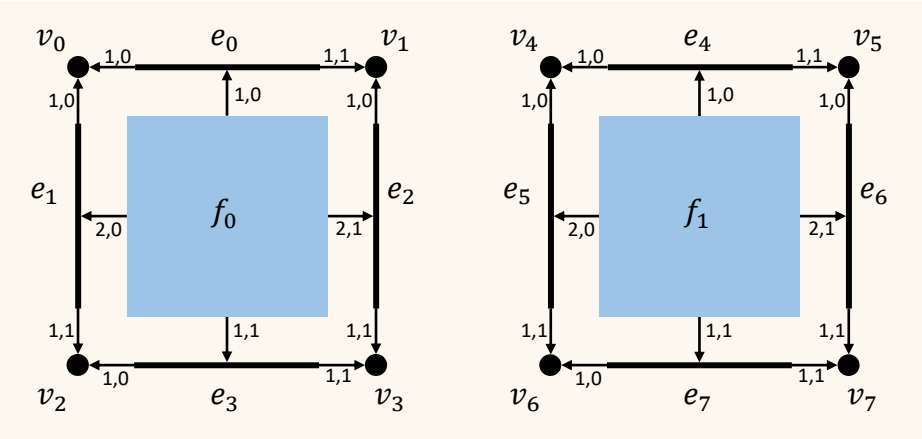
$$I^0 = \{\{v_0\}, \{v_1, v_4\}, \{v_2\}, \{v_3, v_6\}, \{v_5\}, \{v_7\}\}$$
$$I^1 = \{\{e_0\}, \{e_1\}, \{e_2, e_5\}, \{e_3\}, \{e_4\}, \{e_6\}, \{e_7\}\}$$
$$I^2 = \{\{f_0\}, \{f_1\}\}$$

En particulier, $v_1\zeta = v_4\zeta = v_1$, $v_3\zeta = v_6\zeta = v_3$, $e_2\zeta = e_5\zeta = e_2$. L'ensemble gradué K' est défini par :

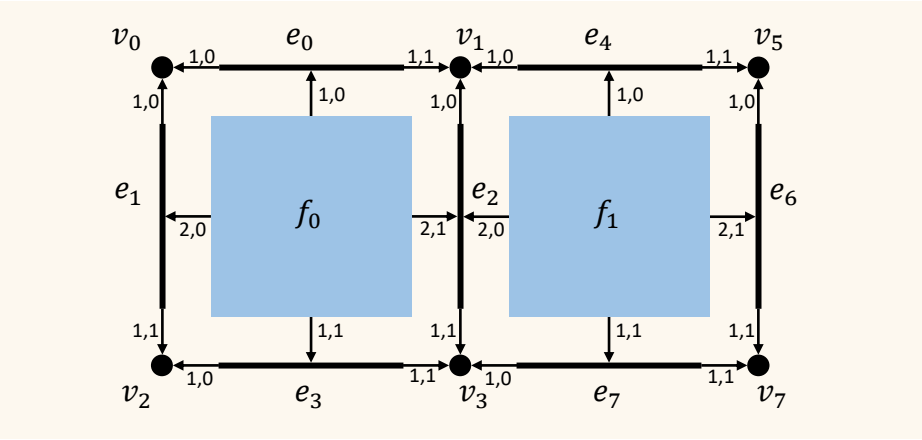
$$K'^0 = \{v_0, v_1, v_2, v_3, v_5, v_7\} = K^0 \setminus \{v_4, v_6\}$$
$$K'^1 = \{e_0, e_1, e_2, e_3, e_4, e_6, e_7\} \setminus \{e_5\}$$
$$K'^2 = \{f_0, f_1\}$$

Notons que :

— $e_4 d_{1,0} = v_4$ et $e_4 d'_{1,0} = v_1$;
— $e_7 d_{1,0} = v_6$ et $e_7 d'_{1,0} = v_3$;
— et $f_1 d_{2,0} = e_5$ et $f_1 d'_{2,0} = e_2$.



(a)



(b)

FIGURE 1.15 – Identification. (a) L'ensemble semi-cubique initial $(K, d)^2$. (b) L'ensemble semi-cubique $(K', d')^2$ résultant de l'identification de v_1 avec v_4, v_3 avec v_6 et e_2 avec e_5.

Désidentification. La désidentification est l'inverse de l'identification. Intuitivement, elle consiste à "éclater" des cubes d'un ensemble semi-cubique et à modifier les opérateurs de face des cubes de l'étoile des cubes éclatés. La figure 1.16 illustre l'application d'une désidentification sur un ensemble semi-cubique (K, d).

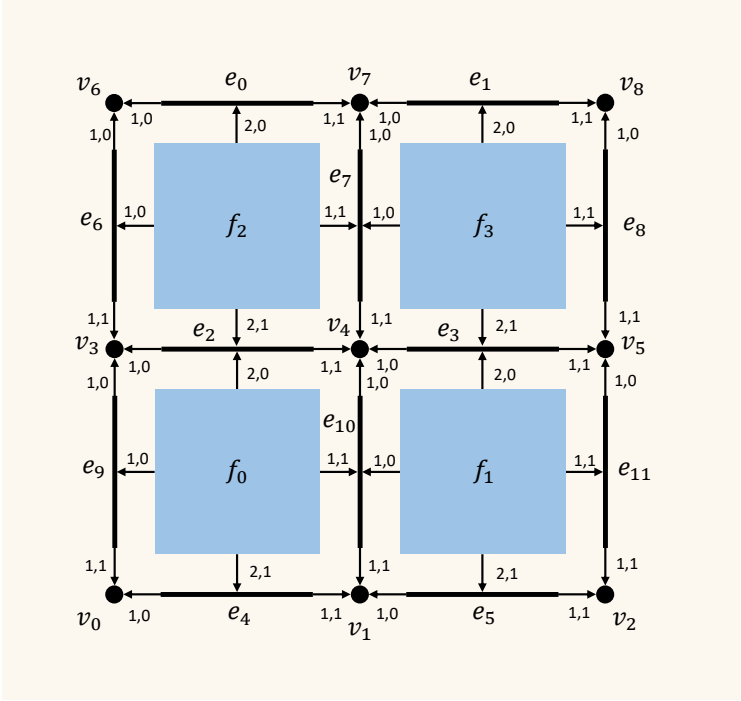
Remarque. Comme pour les ensembles semi-simpliciaux, nous faisons le choix de ne pas donner de définition formelle de la désidentification. Une désidentification est entièrement définie par l'identification inverse qu'elle induit.

1.2.2-b Ensembles semi-simploïdaux

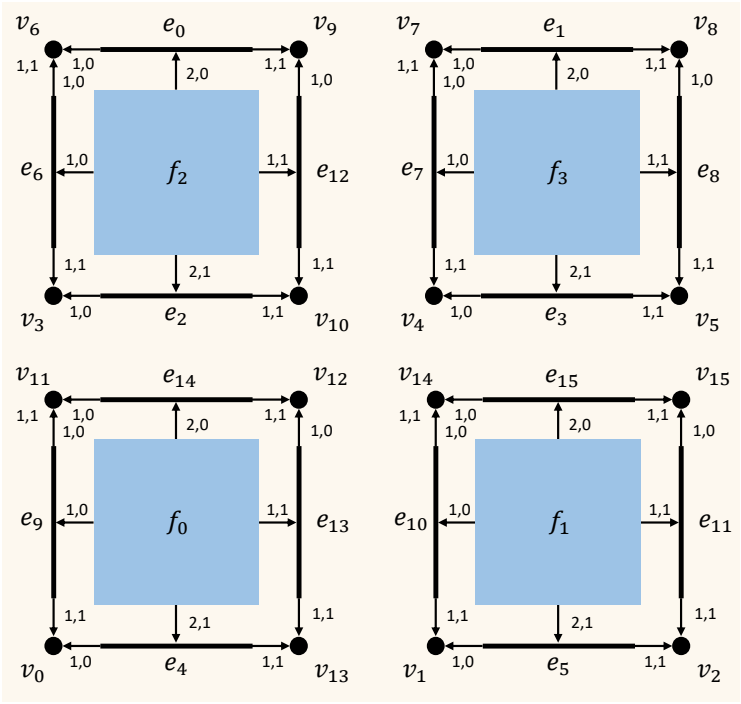
Un **simploïde géométrique** σ est le produit cartésien de simplexes géométriques :

$$\sigma = \sigma_1 \times \sigma_2 \times \ldots \times \sigma_q$$

Le **type** de σ est le q-uplet $(p_1, \ldots, p_q)$ où p_i est la dimension de σ_i pour $1 \leq i \leq q$. La figure 1.17 illustre des simploïdes de différents types.

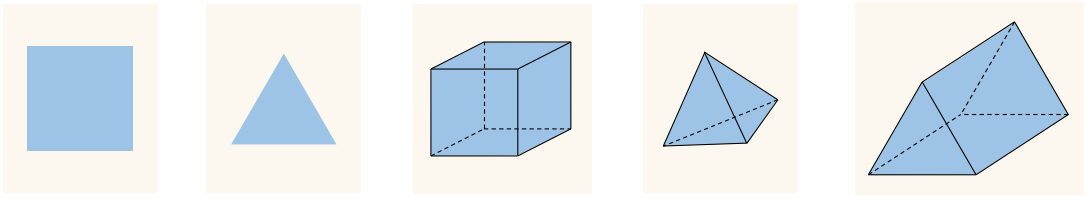


(a)



(b)

FIGURE 1.16 – Désidentification. (a) L'ensemble semi-cubique initial $(K, d)^2$. (b) L'ensemble semi-cubique $(K', d')^2$ résultant d'un éclatement de v_4 en $v_4, v_{10}, v_{12}, v_{14}$, de v_1 en v_1, v_{13}, de v_3 en v_3, v_{11}, de v_7 en v_7, v_9, de v_5 en v_5, v_{15}, de e_7 en e_7, e_{12}, de e_3 en e_3, e_{15}, de e_{10} en e_{10}, e_{13} et de e_2 en e_2, e_{14}.



(a) (b) (c) (d) (e)

FIGURE 1.17 – Représentation graphique de simploïdes de différents types. (a) De type $(1,1)$. (b) De type (2). (c) De type $(1,1,1)$. (d) De type (3). (e) De type $(2,1)$ ou $(1,2)$.

Définition 1.2.14. *Un **ensemble semi-simploidal** de dimension n est un couple $(K,d)^n$ tel que :*

- $K = \bigcup_{0\leq p\leq n} K^p$ *est un ensemble gradué ;*
- $T : K^p \to \bigcup_{q=1}^{p} \mathbb{N}^q$ *vérifie : pour tout p valide et tout $\sigma \in K^p$,*
  - $\sigma T = (0)$ *si* $p = 0$ *;*
  - $\sigma T = (p_1,\dots,p_q) \in \mathbb{N}^{*q}$ *sinon, avec* $p = \sum_{i=1}^{q} p_i$ *;*
- $d : K \to K$ *est une application graduée* $(d^p_{i,j})^{1\leq p\leq n}_{1\leq i\leq q, 0\leq j\leq p_i}$ *de degré -1 vérifiant : pour tout p valide et tout $\sigma \in K^p$, notons* $\sigma T = (p_1,\dots,p_q)$,
  - $\sigma d^p_{i,j} T = \begin{cases} (p_1,\dots,p_i-1,\dots,p_q) \text{ si } p_i > 1 \text{ ;} \\ (p_1,\dots,p_{i-1},p_{i+1},\dots,p_q) \text{ si } p_i = 1 \text{ ;} \end{cases}$
  - $\sigma d^p_{i,j} d^{p-1}_{i,k} = \sigma d^p_{i,k} d^{p-1}_{i,j-1}$ *si* $j > k$ *et* $p_i > 1$ *;*
  - $\sigma d^p_{i,j} d^{p-1}_{k,l} = \begin{cases} \sigma d^p_{k,l} d^{p-1}_{i,j} \text{ si } p_i > 1 \text{ ;} \\ \sigma d^p_{k,l} d^{p-1}_{i-1,j} \text{ sinon,} \end{cases}$ *avec* $k < i$.

Remarque. Les ensembles semi-simploïdaux n'étant pas utilisés par la suite, nous choisissons de ne pas introduire de terminologie. Notons que la terminologie utilisée pour les ensembles semi-simpliciaux et semi-cubiques serait directement applicable ici : un élément de K^p est un p-**simploïde**, une application $d^p_{i,j}$ est appelée **opérateur de face** etc...

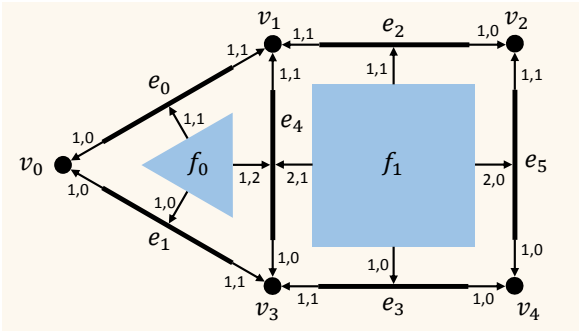


FIGURE 1.18 – Représentation graphique d'un ensemble semi-simploïdal $(K,d)^2$ dans laquelle les opérateurs de face $(d_{i,j})$ sont illustrés par des flèches annotées par i et j.

Exemple. La figure 1.18 illustre un ensemble semi-simploïdal $(K, d)^2$. Les simploïdes de K sont :

$$K^0 = \{v_0, v_1, v_2, v_3, v_4\}$$
$$K^1 = \{e_0, e_1, e_2, e_3, e_4, e_5\}$$
$$K^2 = \{f_0, f_1\}$$

En particulier :

— $f_0 T = (2)$ et $f_0 d_{1,0} T = f_0 d_{1,1} T = f_0 d_{1,2} T = (1)$;
— $f_1 T = (1, 1)$ et $f_1 d_{1,0} T = f_1 d_{1,1} T = f_1 d_{2,1} T = f_1 d_{2,0} T = (1)$;
— $f_0 d_{1,2} d_{1,1} = f_0 d_{1,1} d_{1,1}$;
— $f_1 d_{2,1} d_{1,1} = f_1 d_{1,1} d_{1,1}$

1.2.3 Structures cellulaires

Les structures cellulaires permettent de représenter des objets subdivisés en cellules "quelconques" (cf figure 1.19). Nous présentons dans cette sous-section les ensembles semi-simpliciaux numérotés. À notre connaissance, toutes les structures cellulaires sont des représentations optimisées de ces derniers, en particulier :

— les graphes d'incidences ;
— les cartes combinatoires, utilisées dans le Chapitre 3.

Les définitions et la terminologie de cette sous-section s'inspirent de [24].

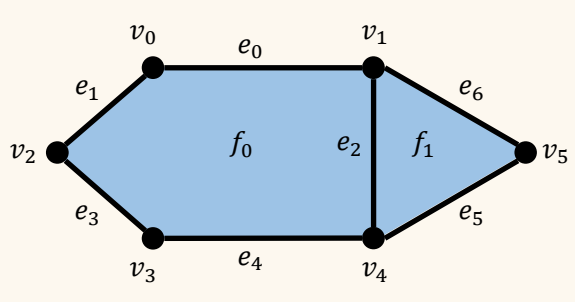
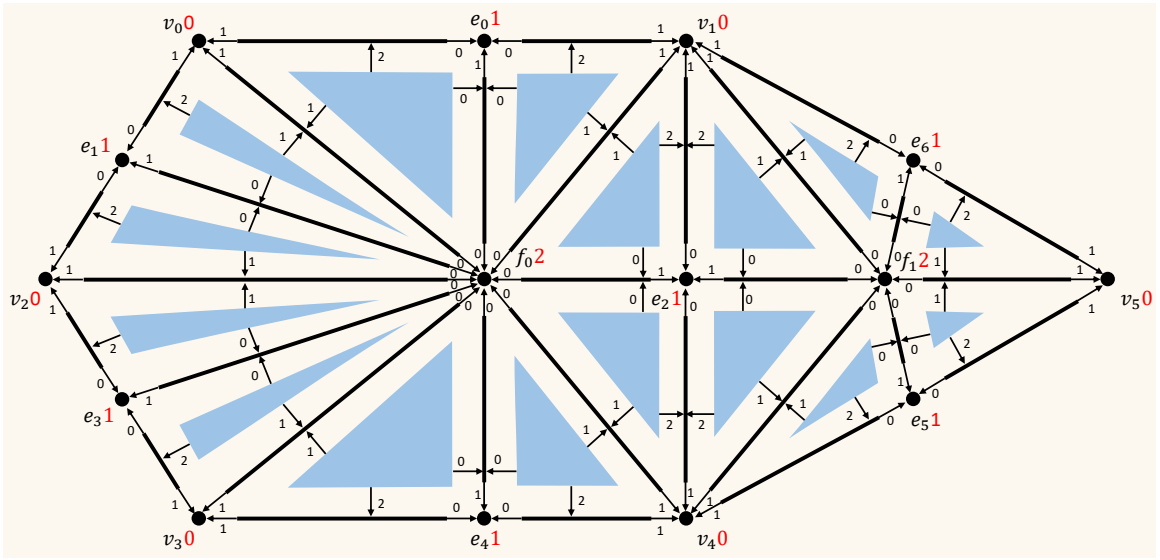


FIGURE 1.19 – Représentation graphique d'un objet subdivisé en cellules quelconques, composé de 6 sommets, 7 arêtes et 2 faces.

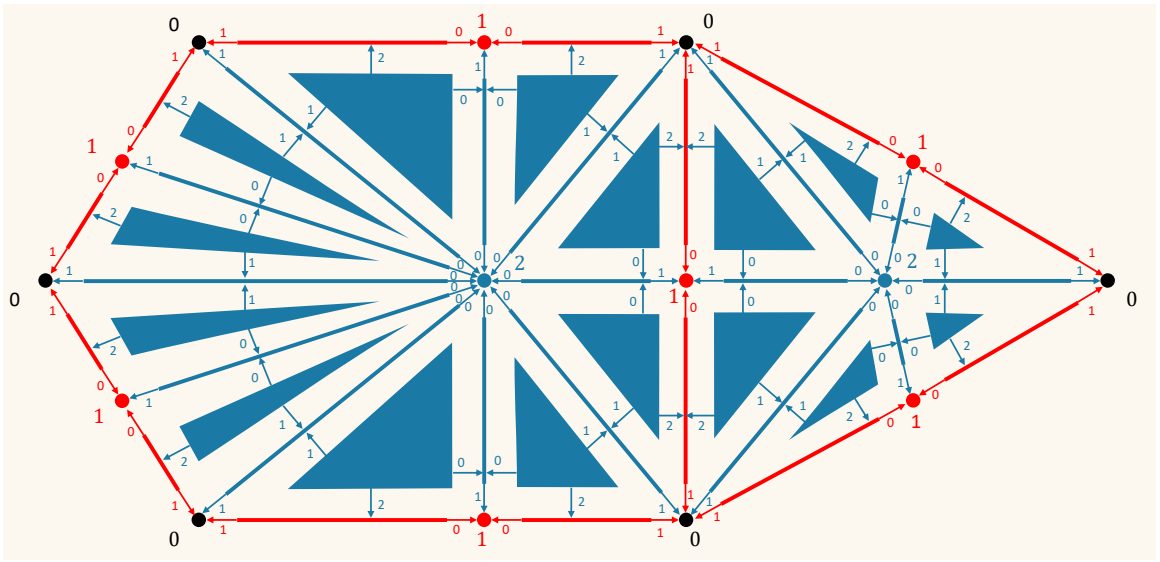
1.2.3-a Ensembles semi-simpliciaux numérotés

Définition 1.2.15. *Un **ensemble semi-simplicial numéroté** est un ensemble semi-simplicial (K, d) muni d'une application $N : K^0 \to \mathbb{N}$ telle que, pour tout $\sigma \in K^p$, les sommets incidents à σ ont des numéros distincts.*

Définition 1.2.16. *Une **p-cellule** est composée d'un sommet numéroté p et de son étoile restreinte aux simplexes de dimensions inférieures ou égales à p incidents à des sommets dont les numéros sont inférieurs ou égaux à p.*



(a)

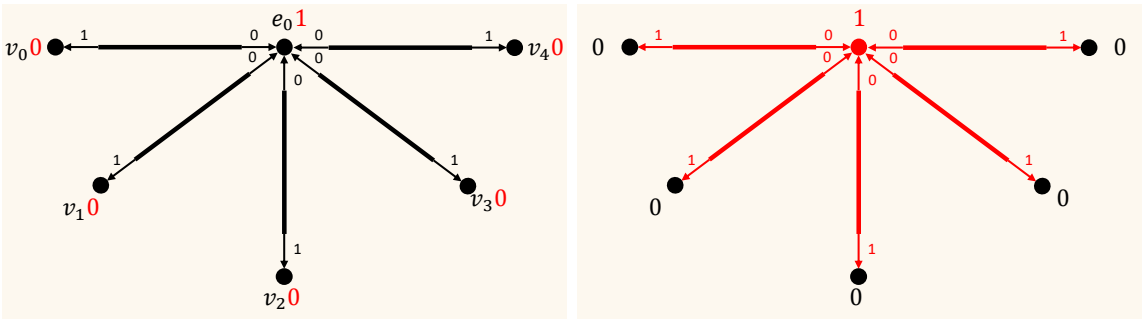


(b)

FIGURE 1.20 – (a) Représentation graphique d'un ensemble semi-simplicial numéroté (K, d). En rouge, la numérotation associée aux 0-simplexes. (b) En bleu, les 2-cellules (il y en a 2), en rouge, les 1-cellules (il y en a 7), en noir, les 0-cellules (il y en a 6).

Exemple. La figure 1.20 illustre un ensemble semi-simplicial numéroté et les cellules induites.

Remarque. On s'intéresse généralement à des sous classes d'ensembles semi-simpliciaux numérotés pour éviter, par exemple, de manipuler des cellules comme celle de la figure 1.21.



(a) (b)

FIGURE 1.21 – (a) Représentation graphique d'un ensemble semi-simplicial numéroté. (b) En rouge, la 1-cellule, en noir, les 0-cellules. Notons que la 1-cellule a 5 sommets dans son bord.

1.2.3-b Graphes d'incidences.

Un graphe d'incidence est composé d'un ensemble **d'arcs** A et d'un ensemble de **nœuds** V. Un nœud représente une cellule. Un arc représente une relation d'incidence entre deux cellules.

Remarque. Il existe différentes variantes de graphes d'incidence : on choisit la définition ci-dessous pour illustrer cette notion.

Définition 1.2.17. *Un **graphe d'incidence** $G = (V, A)^n$ est composé d'un ensemble gradué $V = \bigcup_{p=0}^{n} V^p$ et d'un ensemble $A \subset \bigcup_{p=1}^{n} (V^{p-1} \times V^p)$.*

À tout graphe d'incidence $G = (V, A)$ ainsi défini peut être associé un ensemble semi-simplicial numéroté (K, d) de la manière suivante :

— un simplexe σ de dimension q est associé à $(c_{x_0}, \ldots, c_{x_q}) \in V^{x_0} \times \ldots \times V^{x_q}$ s'il existe un chemin $(c_0, \ldots, c_n)$ dont $(c_{x_0}, \ldots, c_{x_q})$ est un sous-chemin. Pour tout i compris entre 0 et q, σd_i est le simplexe associé au sous-chemin $(c_{x_0}, \ldots, c_{x_{i-1}}, c_{x_{i+1}}, \ldots c_{x_q})$;

— tout 0-simplexe associé à un sommet de V^p est numéroté p.

Remarque. À tout graphe d'incidence peut être associé un ensemble semi-simplicial numéroté, l'inverse n'est pas vrai. Par exemple, l'ensemble semi-simplicial numéroté de la figure 1.20 peut être associé au graphe d'incidence de la figure 1.22, mais on ne peut pas associer de graphe d'incidence à l'ensemble semi-simplicial numéroté de la figure 1.23 car il faudrait deux chemins (f_0, e_0, v_0) pour représenter les 2 simplexes de dimension 2.

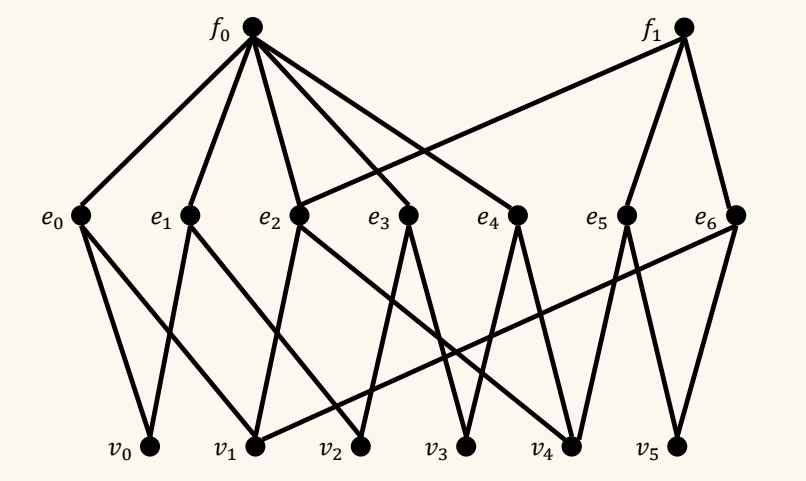
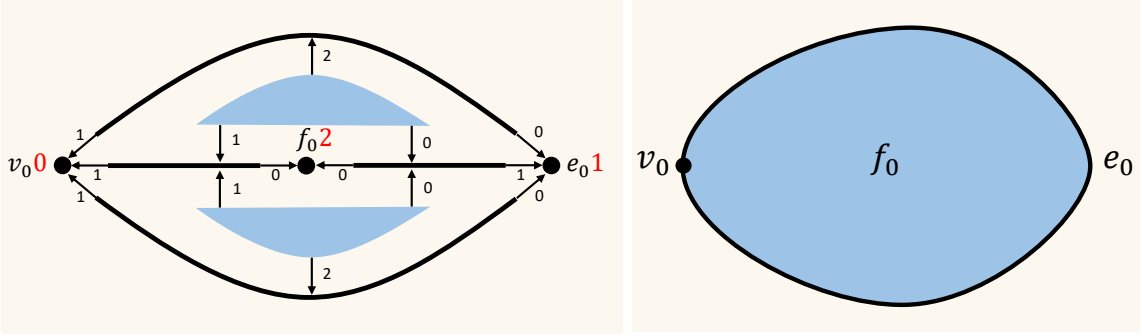


FIGURE 1.22 – Représentation graphique du graphe d'incidence de l'objet subdivisé de la figure 1.19.

Exemple. Le graphe d'incidence de la figure 1.22 est défini par :

$$V^0 = \{v_0, v_1, v_2, v_3, v_4, v_5\}$$
$$V^1 = \{e_0, e_1, e_2, e_3, e_4, e_5, e_6\}$$
$$V^2 = \{f_0, f_1\}$$
$$A = \left\{ \begin{array}{c} (v_0, e_0), (v_0, e_1), (v_1, e_0), (v_1, e_6), (v_2, e_1), (v_2, e_3), (v_3, e_3), (v_3, e_4) \\ (v_4, e_2), (v_4, e_4), (v_4, e_5), (v_5, e_5), (v_5, e_6) \\ (e_0, f_0), (e_1, f_0), (e_2, f_0), (e_2, f_1), (e_3, f_0), (e_4, f_0), (e_5, f_1), (e_6, f_1) \end{array} \right\}$$



(a) (b)

FIGURE 1.23 – Un ensemble semi-simplicial numéroté (K, d) auquel on ne peut pas associer de graphe d'incidence tel que défini dans cette sous-section. (a) (K, d), la numérotation des sommets est en rouge. (b) Un objet géométrique pouvant être associé à (K, d).

En particulier, l'arc (v_1, e_0) indique que v_1 est incident à e_0.

1.2.3-c Cartes combinatoires.

Il existe différentes variantes de cartes combinatoires. On choisit ici de présenter les *cartes généralisées*, telles que définies dans [25].

Définition et terminologie.

Définition 1.2.18. *Une **carte généralisée** de dimension n est un couple $(B, \alpha)^n$, où*

- *B est un ensemble ;*
- *$\alpha : B \to B$ est une famille d'involutions*[4] *$(\alpha_i)^{0 \leq i \leq n}$ telle que :*
  - *$\alpha_i : B \to B$;*
  - *$\alpha_i \alpha_j$ est une involution,* $\forall i, j | 0 \leq i < i + 2 \leq j \leq n.$

Un élément de B est appelé un **brin**. L'ensemble de tous les brins accessibles à partir de b pour une combinaison quelconque d'involutions α_{i_0}, ..., α_{i_q} est une **orbite**, notée $< \alpha_{i_0}, \ldots, \alpha_{i_q} > (b)$.

Lien avec les ensembles semi-simpliciaux numérotés. Les cartes généralisées peuvent être associées à une sous-classe d'ensembles semi-simpliciaux numérotés, les *quasi-variétés cellulaires* (non définies ici, cf [25] section 2.2). Intuitivement :

- une quasi-variété cellulaire de dimension 0 est composée d'une ou deux cellules de dimension 0 ;
- une cellule de dimension n est un cône sur une quasi-variété cellulaire connexe[5] de dimension $n - 1$;

4. Une **involution** est une application $\alpha : B \to B$ qui est son propre inverse, c'est-à-dire que $b\alpha\alpha = b$ pour tout b de B.

5. Une quasi-variété cellulaire de dimension 0 composée de deux cellules est connexe (une sphère de dimension 0 est composée de deux sommets).

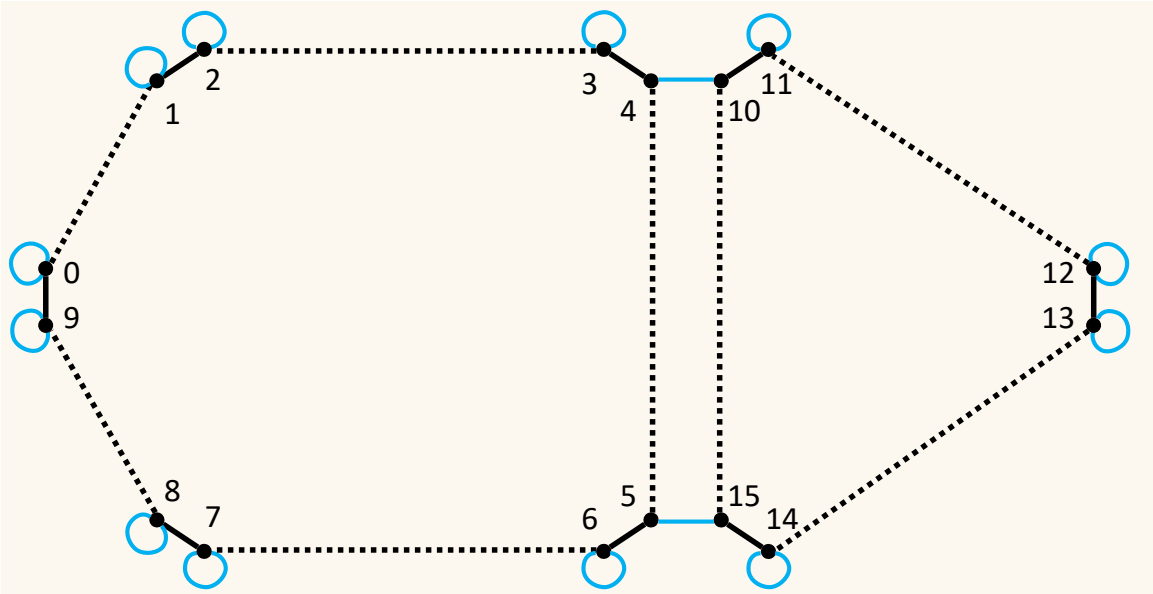


FIGURE 1.24 – Représentation graphique de la carte généralisée $(B, \alpha)^2$ associée à l'objet subdivisé de la figure 1.19. Elle est composée de 16 brins, représentés par des sommets. L'involution α_0 est représentée par un trait en pointillé, l'involution α_1 par un trait plein, l'involution α_2 par un trait bleu.

— une quasi-variété cellulaire de dimension n est construite par identifications de cellules de dimension n le long de cellules de dimension $(n-1)$ impliquant au plus 2 cellules de dimension $(n-1)$.

Soient une carte généralisée $(B, \alpha)^n$ et $N = \{0, \ldots, n\}$. L'ensemble semi-simplicial numéroté (K, d) induit par $(B, \alpha)^n$ est défini par : pour tout $I = \{i_0, \ldots, i_p\}$ et tout $b \in B$,

— un p-simplexe est associé à l'orbite

$$<>_{N-I}(b) =< \alpha_0, \ldots, \hat{\alpha_{i_0}}, \ldots, \hat{\alpha_{i_p}}, \ldots, \alpha_n > (b) \qquad \forall p, n | 0 \leq p \leq n$$

où $\hat{\alpha_i}$ indique que α_i ne fait pas partie de l'orbite. Le 0-simplexe associé à l'orbite $<>_{N-\{i\}}(b)$ est numéroté i ;

— soit σ le p-simplexe associé à l'orbite $<>_{N-I}(b)$. σd_j est le $(p-1)$-simplexe associé à l'orbite $<>_{N-I-\{i_j\}}$ pour tout j compris entre 0 et p.

Remarque. Par la suite, notamment dans le chapitre 3, nous utilisons la notion de carte orientée. Une carte orientée permet de représenter des quasi-variétés cellulaires orientées (cf. [26]).

Exemple. L'ensemble de brins de la carte généralisée $(B, \alpha)^2$ illustrée sur la figure 1.24 est :

$$B = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$$

Notons par exemple que $\alpha_0\alpha_2$ est une involution. En particulier $4\alpha_0\alpha_2 = 15$ et $15\alpha_0\alpha_2 = 4$. L'ensemble semi-simplicial numéroté associé à $(B, \alpha)^2$ est représenté sur la figure 1.20. Pour simplifier, un simplexe est dénoté par la suite de ses sommets incidents. Soit $N = \{0, 1, 2\}$:

— le 0-simplexe (f_0), numéroté 2, est associé à l'orbite $<>_{N-\{2\}}(2) = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$;

— le 0-simplexe (e_0), numéroté 1, est associé à l'orbite $<>_{N-\{1\}}(2) = \{2,3\}$;
— le 0-simplexe (v_0), numéroté 0, est associé à l'orbite $<>_{N-\{0\}}(2) = \{1,2\}$;
— le 1-simplexe (e_0, f_0) est associé à l'orbite $<>_{N-\{1,2\}}(2) = \{2,3\}$;
— le 1-simplexe (v_0, f_0) est associé à l'orbite $<>_{N-\{0,2\}}(2) = \{1,2\}$;
— le 1-simplexe (v_0, e_0) est associé à l'orbite $<>_{N-\{0,1\}}(2) = \{2\}$;
— le 2-simplexe (v_0, e_0, f_0) est associé à l'orbite $<>_{N-\{0,1,2\}}(2) = \{2\}$;

Considérons le 2-simplexe (v_0, e_0, f_0) :

— $(v_0, e_0, f_0)d_2 = (v_0, e_0)$, associé à l'orbite $<>_{N-\{0,1,2\}-\{2\}}(2)$;
— $(v_0, e_0, f_0)d_1 = (v_0, f_0)$, associé à l'orbite $<>_{N-\{0,1,2\}-\{1\}}(2)$;
— $(v_0, e_0, f_0)d_0 = (e_0, f_0)$, associé à l'orbite $<>_{N-\{0,1,2\}-\{0\}}(2)$; ;

1.3 Groupes d'homologie

Les groupes d'homologie caractérisent une partie de l'information structurelle d'un objet, par exemple son nombre de composantes connexes.

Remarque. Les documents de référence pour cette section sont [16] et [1]. Dans les deux cas, les auteurs ont choisi de travailler avec des *modules* et non des groupes abéliens. Ceci implique de définir la totalité des notions qui suivent sur des *modules*, y compris la notion de "groupes" d'homologie. D'après la propriété 1.1, un groupe abélien est un *module* particulier ; par souci de clarté, nous avons choisi d'adapter les définitions des documents de référence aux groupes abéliens.

1.3.1 Complexes de chaînes

Dans cette sous-section, nous définissons les complexes de chaînes, dont sont déduits les groupes d'homologie.

Définition 1.3.1. *Un **complexe de chaînes** de dimension $n \in \mathbb{N}^*$ est un couple $(C, \partial)^n$, où :*

— *C est un groupe abélien gradué ;*
— *$\partial : C \to C$ est un morphisme gradué de degré -1 vérifiant :*

$$\partial^p \partial^{p-1} = 0 \qquad \forall p | 1 \leq p \leq n$$

Notons qu'en toute rigueur, d'après la définition 1.3.1, ∂^0 n'est pas défini. L'extension $\partial^0 : C^0 \to \{0\}$ est usuellement utilisée dans la définition des complexes de chaînes. Un complexe de chaînes $(C, \partial)^n$ est représenté par :

$$C^n \xrightarrow{\partial^n} C^{n-1} \xrightarrow{\partial^{n-1}} \dots \xrightarrow{\partial^2} C^1 \xrightarrow{\partial^1} C^0 \xrightarrow{\partial^0} 0$$

Un élément de C^p est une p-**chaîne**. Le morphisme gradué ∂ est l'**opérateur de bord**.

Remarque. Par la suite, les exposants des opérateurs de bord et des complexes de chaînes sont parfois omis.

Définition 1.3.2. *Soient deux complexes de chaînes (C, ∂) et (C', ∂'). Un **morphisme de complexe de chaînes** de (C, ∂) sur (C', ∂') est un morphisme gradué de (C, ∂) sur (C', ∂'). En particulier, $\phi : C \to C'$ vérifie :*

$$\partial^p \phi^{p-1} = \phi^p \partial'^p \qquad \forall p | 1 \leq p \leq n$$

1.3.2 De la structure topologique aux complexes de chaînes

Dans cette sous-section, nous présentons une méthode de construction d'un complexe de chaînes à partir d'une structure topologique pour les ensembles semi-simpliciaux, les ensembles semi-cubiques et les cartes généralisées.

1.3.2-a Ensembles semi-simpliciaux

Ce paragraphe est une adaptation de [16] page 16. On peut associer à tout ensemble semi-simplicial (K, d) un complexe de chaînes (C, ∂). Informellement :

— les générateurs[6] de C sont les simplexes de K ;

— le bord de chaque générateur correspond au bord du simplexe auquel il est associé.

Définition 1.3.3. *Soit un ensemble semi-simplicial (K, d). On peut lui associer le complexe de chaînes (C, ∂) tel que :*

— *C est engendré par K ;*

— *∂ est défini par :*

· $\sigma\partial^0 = 0 \qquad \forall \sigma \in K^0$;

· $\sigma\partial^p = \sum_{i=0}^{p}(-1)^i \sigma d_i \qquad \forall \sigma \in K^p, \forall p | p > 0$

La convention choisie pour la représentation graphique du complexe de chaînes (C, ∂) associé à un ensemble semi-simplicial (K, d) est la suivante :

— seuls les générateurs de C sont représentés ;

— l'opérateur de bord est représenté par une flèche :

· allant de d_1 vers d_0 en dimension 1 ;

· passant par d_0, puis d_1, puis d_2 en dimension 2. Plus précisément, l'orientation de la face respecte l'orientation des arêtes de son bord en tenant compte des coefficients $+1$ ou -1 (cf figure 1.25).

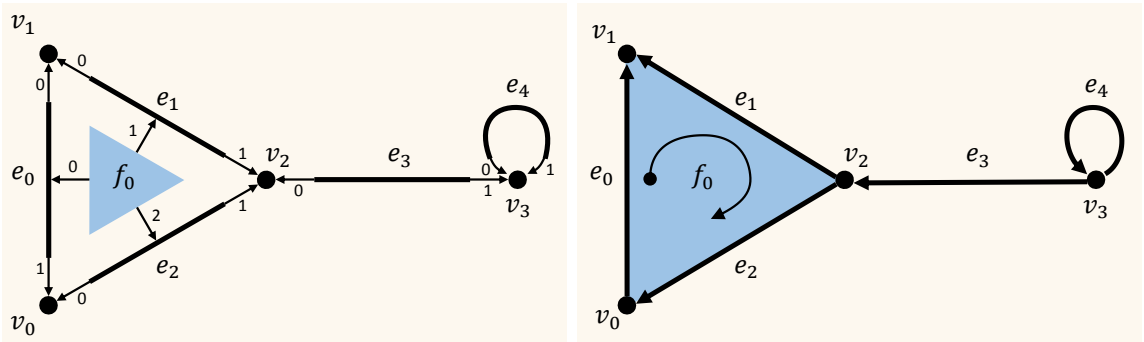


FIGURE 1.25 – Représentation graphique d'un ensemble semi-simplicial $(K, d)^2$ et du complexe de chaînes $(C, \partial)^2$ qui lui est associé. (a) L'ensemble semi-simplicial $(K, d)^2$. (b) Le complexe de chaînes $(C, \partial)^2$.

6. Les générateurs d'un groupe C sont les éléments d'une partie P de C telle que tout élément de C s'écrit comme une combinaison d'éléments de P.

Exemple. Les générateurs du complexe de chaînes $(C, \partial)^2$ illustré sur la figure 1.25b sont :

$$C^0 = \{v_0, v_1, v_2, v_3\}$$
$$C^1 = \{e_0, e_1, e_2, e_3, e_4\}$$
$$C^2 = \{f_0\}$$

En particulier, d'après la convention graphique établie, les flèches de la figure indiquent que :

— $f_0\partial^2 = e_0 - e_1 + e_2$;

— $e_0\partial^1 = v_1 - v_0$, $e_1\partial^1 = v_1 - v_2$, $e_2\partial^1 = v_0 - v_2$, $e_3\partial^1 = v_2 - v_3$ et $e_4\partial^1 = v_3 - v_3 = 0$.

1.3.2-b Ensembles simpliciaux

On peut associer à tout ensemble simplicial (K, d, s) un complexe de chaînes (C, ∂).

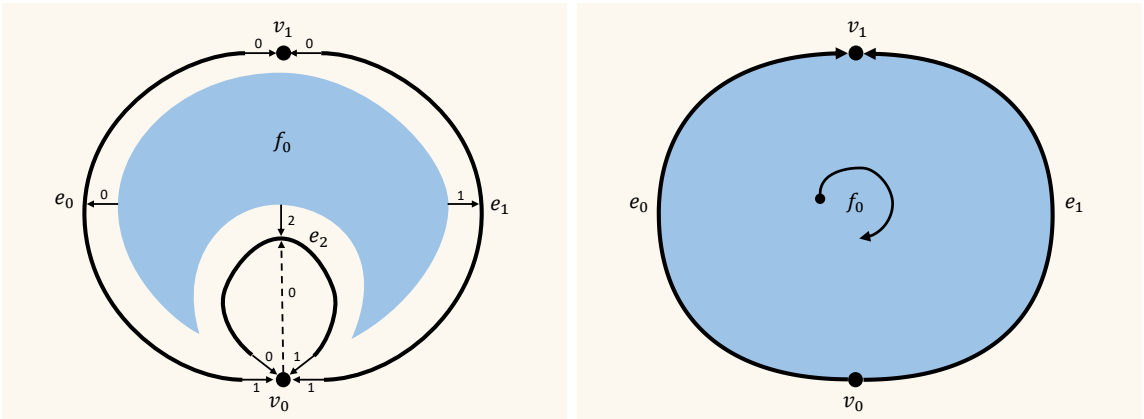
Définition 1.3.4. *Soit un ensemble simplicial (K, d, s). Le complexe de chaînes (C, ∂) qui lui est associé est tel que :*

— *C est engendré par les simplexes non dégénérés de K ;*

— *∂ est défini par :*

· $\sigma\partial^0 = 0 \qquad \forall \sigma \in K^0$;

· $\sigma\partial^p = \sum_{i=0}^{p} \alpha_i (-1)^i \sigma d_i$ *où* $\alpha_i = 0$ *si* σd_i *est dégénéré et* $\alpha_i = 1$ *sinon.*



(a) (b)

FIGURE 1.26 – Représentation graphique du complexe de chaînes associé à un ensemble simplicial. (a) L'ensemble simplicial. (b) Le complexe de chaînes.

Exemple. Les générateurs du complexe de chaînes $(C, \partial)^2$ illustré sur la figure 1.26 sont :

$$C^0 = \{v_0, v_1\}$$
$$C^1 = \{e_0, e_1\}$$
$$C^2 = \{f_0\}$$

En particulier, d'après la convention graphique établie, les flèches de la figure indiquent que :

— $f_0\partial^2 = e_0 - e_1$;

— $e_0\partial^1 = v_1 - v_0$, $e_1\partial^1 = v_1 - v_0$.

1.3.2-c Ensembles semi-cubiques

Ce paragraphe est une adaptation de [27] page 181. On peut associer à tout ensemble semi-cubique (K, d) un complexe de chaînes (C, ∂). Informellement :

— les générateurs de C sont les cubes de K ;

— le bord de chaque générateur correspond au bord du cube auquel il est associé ;

Définition 1.3.5. *Soit un ensemble semi-cubique (K, d). On peut lui associer le complexe de chaînes (C, ∂) tel que :*

— *C est engendré par K ;*

— *∂ est défini par :*

· $\sigma\partial^0 = 0 \qquad \forall \sigma \in K^0$;

· $\sigma\partial^p = \sum_{i=1}^{p}(-1)^i(\sigma d^p_{i,1} - \sigma d^p_{i,0}) \qquad \forall \sigma \in K^p, \forall p | p > 0$

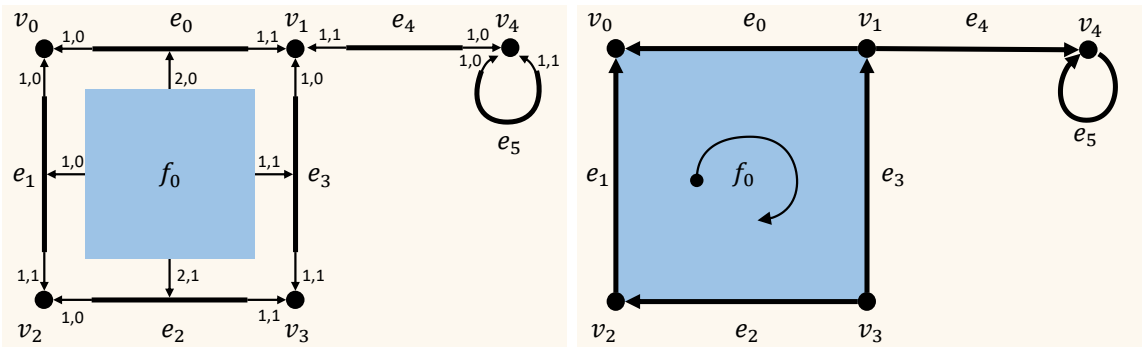
La convention choisie pour la représentation graphique du complexe de chaînes (C, ∂) associé à un ensemble semi-cubique (K, d) est la suivante :

— seuls les générateurs de C sont représentés ;

— l'opérateur de bord est représenté par une flèche :

· allant de $d_{1,1}$ vers $d_{1,0}$ en dimension 1 ;

· passant par $d_{1,0}$, puis $d_{2,0}$, puis $d_{1,1}$, puis $d_{2,1}$ en dimension 2. Plus précisément, l'orientation de la face respecte l'orientation des arêtes de son bord en tenant compte des coefficients $+1$ ou -1 (cf figure 1.27).



(a) (b)

FIGURE 1.27 – Représentation graphique d'un ensemble semi-cubique $(K, d)^2$ et du complexe de chaînes $(C, \partial)^2$ qui lui est associé. (a) L'ensemble semi-cubique $(K, d)^2$. (b) Le complexe de chaînes $(C, \partial)^2$.

Exemple. Les générateurs du complexe de chaînes (C, ∂) illustré sur la figure 1.27b sont :

$$C^0 = \{v_0, v_1, v_2, v_3, v_4\}$$
$$C^1 = \{e_0, e_1, e_2, e_3, e_4, e_5\}$$
$$C^2 = \{f_0\}$$

En particulier, d'après la convention graphique établie, les flèches de la figure indiquent que :

— $f_0\partial^2 = -(e_3 - e_1) + (e_2 - e_0)$
— $e_0\partial^1 = v_0 - v_1$, $e_1\partial^1 = v_0 - v_2$, $e_2\partial^1 = v_2 - v_3$ et $e_3\partial^1 = v_1 - v_3$, $e_4\partial^1 = v_4 - v_1$ et $e_5\partial^1 = v_4 - v_4 = 0$

1.3.2-d Cartes généralisées

Il existe plusieurs façons de construire un complexe de chaînes (C, ∂) à partir d'une carte généralisée (B, α) :

1. en associant un ensemble semi-simplicial numéroté (K, d) à (B, α) (cf paragraphe 1.2.3-c), puis en construisant le complexe de chaînes que l'ensemble semi-simplicial induit (cf paragraphe 1.3.2-a). Les générateurs de C sont alors les simplexes de K. Dans ce cas, on oublie la structuration des simplexes en cellules. Il est néanmoins possible d'en tirer parti pour optimiser les calculs de groupes d'homologie, en mémorisant les informations liées à l'homologie des cellules (cf section 4 de [16]) ;
2. en associant directement à chaque cellule de (B, α) un générateur dans (C, ∂), et en déduisant ∂ des relations entre les cellules de la carte. Ceci n'est possible sans perte d'information homologique que dans le cas où les groupes d'homologie des bords des cellules sont isomorphes à ceux d'une sphère, ce qui est généralement le cas en pratique (cf [28]).

Remarque. L'objectif de ce paragraphe est de mettre en évidence les critères sur lesquels choisir une méthode de construction plutôt qu'une autre. Ces dernières sont donc présentées succinctement. Nous invitons le lecteur à se référer aux documents cités s'il souhaite approfondir le sujet.

1.3.3 Cycles, bords et groupes d'homologie

Dans cette sous-section, nous définissons les groupes d'homologie, calculés à partir d'un complexe de chaînes, en distinguant deux types de chaînes : les cycles et les bords.

1.3.3-a Cycles et bords

Définition 1.3.6. *Soit un complexe de chaînes (C, ∂). Une p-chaîne $z \in C^p$ est un p-**cycle** si et seulement si $z\partial^p = 0$.*

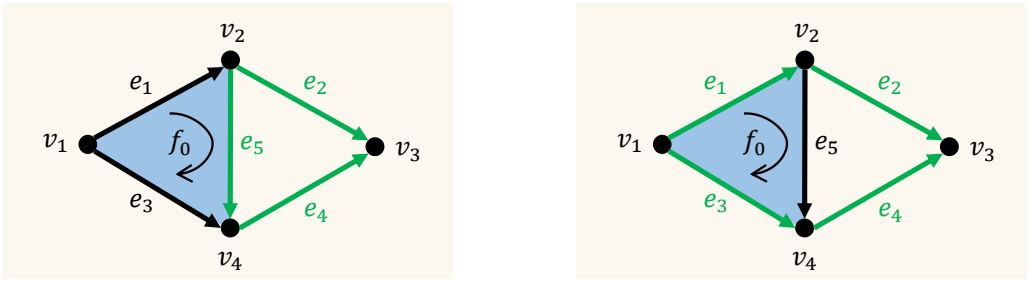
Les p-cycles forment un groupe abélien noté Z^p.

Remarque. Notons que toute 0-chaîne est un 0-cycle car $\partial^0 = 0$.

Exemple. Quelques 1-cycles sont représentés sur la figure 1.28. En particulier, z^a vérifie :

$$\begin{aligned} z^a\partial^1 &= (e_5 + e_4 - e_2)\partial^1 \\ &= (v_4 - v_2) + (v_3 - v_4) - (v_3 - v_2) \\ &= 0 \end{aligned}$$

Définition 1.3.7. *Soit un complexe de chaînes (C, ∂). Une p-chaîne $b \in C^p$ est un p-**bord** si et seulement si il existe une chaîne c dans C^{p+1} telle que $c\partial^{p+1} = b$.*

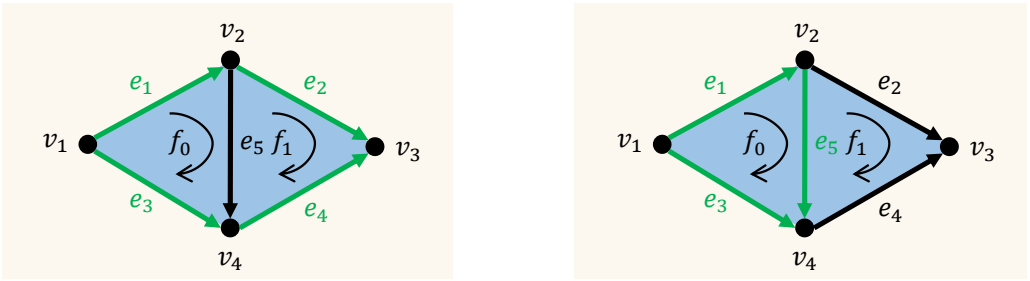


(a) (b)

FIGURE 1.28 – Différents 1-cycles d'un même complexe de chaînes. (a) La 1-chaîne $z^a = e_5 + e_4 - e_2$ est un 1-cycle (en vert). (b) La 1-chaîne $z^b = e_3 + e_4 - e_2 - e_1$ est un 1-cycle (en vert).

Les p-bords forment un groupe abélien noté B^p.

Remarque. Notons que tout bord est un cycle car $\partial\partial = 0$.



(a) (b)

FIGURE 1.29 – Différents 1-bords d'un même complexe de chaînes. (a) La 1-chaîne $b^a = e_1 + e_2 - e_4 - e_3$ est un 1-bord (en vert). (b) La 1-chaîne $b^b = e_1 + e_5 - e_3$ est un 1-bord (en vert).

Exemple. Quelques 1-bords sont illustrés sur la figure 1.29. En particulier :

$$\begin{aligned}(f_0 + f_1)\partial^2 &= (e_1 + e_5 - e_3) + (e_2 - e_4 - e_5) \\ &= e_1 - e_3 + e_2 - e_4 \\ &= b^a\end{aligned}$$

Donc b^a est un 1-bord.

1.3.3-b Groupes d'homologie

Les groupes d'homologies caractérisent les "trous" d'un complexe de chaînes. Un "trou" est une notion abstraite qui n'a pas nécessairement d'interprétation intuitive, d'autant plus lorsque ces derniers sont de dimension supérieure à 2. En supposant que le complexe de chaînes soit associé à un maillage 3D, ses groupes d'homologie caractérisent :

— ses composantes connexes en dimension 0 ;

— ses tunnels en dimension 1 ;

— ses cavités en dimension 2.

Définition et terminologie. Les groupes d'homologie sont calculés à partir d'un complexe de chaînes (C, ∂) comme étant le quotient de ses cycles par ses bords. Informellement, cela signifie que l'on partitionne les cycles de (C, ∂) selon la relation d'équivalence : "deux cycles sont équivalents s'ils ne diffèrent que d'un bord".

Définition 1.3.8. *Soit un complexe de chaînes (C, ∂). **Son p-ème groupe d'homologie** est le groupe quotient :*

$$H^p = Z^p / B^p$$

Deux p-cycles z_0 et z_1 appartenant à une même classe d'équivalence de H^p sont dit **homologues**.

Remarque. Il existe des travaux de recherche autour du calcul des "meilleurs" générateurs des groupes d'homologie selon des critères bien définis [29, 30, 31, 32].

Exemple. Les 1-cycles z^a et z^b illustrés sur la figure 1.28 sont homologues :

$$\begin{aligned} z^a - z^b &= (e_5 + e_4 - e_2) - (e_3 + e_4 - e_2 - e_1) \\ &= e_5 - e_3 + e_1 \\ &= f_0 \partial^2 \end{aligned}$$

Dit autrement, z^a et z^b ne diffèrent que d'un 1-bord $f_0\partial^2$.

Théorème de structure des groupes abéliens de type fini. Dans le cadre de cette thèse, on suppose que les complexes de chaînes sont toujours associés à une structure topologique composée d'un nombre fini de cellules, et que les générateurs des chaînes sont ces cellules. En conséquence, les groupes d'homologie considérés sont des groupes abéliens de type fini et sont sujets au théorème de structure des groupes abéliens de type fini. La présentation du théorème ci-dessous est issue de [20] page 36. Il est présenté plus en détail dans [33] pages 284-285.

Théorème 1. *Tout groupe abélien de type fini est isomorphe à une somme directe finie de groupes monogènes*[7] *:*

$$\underbrace{\mathbb{Z} \oplus \ldots \oplus \mathbb{Z}}_{\text{partie libre}} \oplus \underbrace{\mathbb{Z}/t_1\mathbb{Z} \oplus \ldots \oplus \mathbb{Z}/t_k\mathbb{Z}}_{\text{partie de torsion}}$$

avec $1 < t_i$ et t_i divise t_{i+1} pour tout $t_i \in \mathbb{Z}$, $1 \leq i < k$.

En particulier, les groupes d'homologie $H = (H^p)$ d'un complexe de chaînes sont isomorphes à une somme directe finie de groupes monogènes et se décomposent en une partie libre et une partie de torsion. Le rang de la partie libre de H^p est appelé **nombre de Betti**, et noté β^p. Les entiers t_i de la partie de torsion sont les **coefficients de torsion**.

1.3.4 Quelques exemples

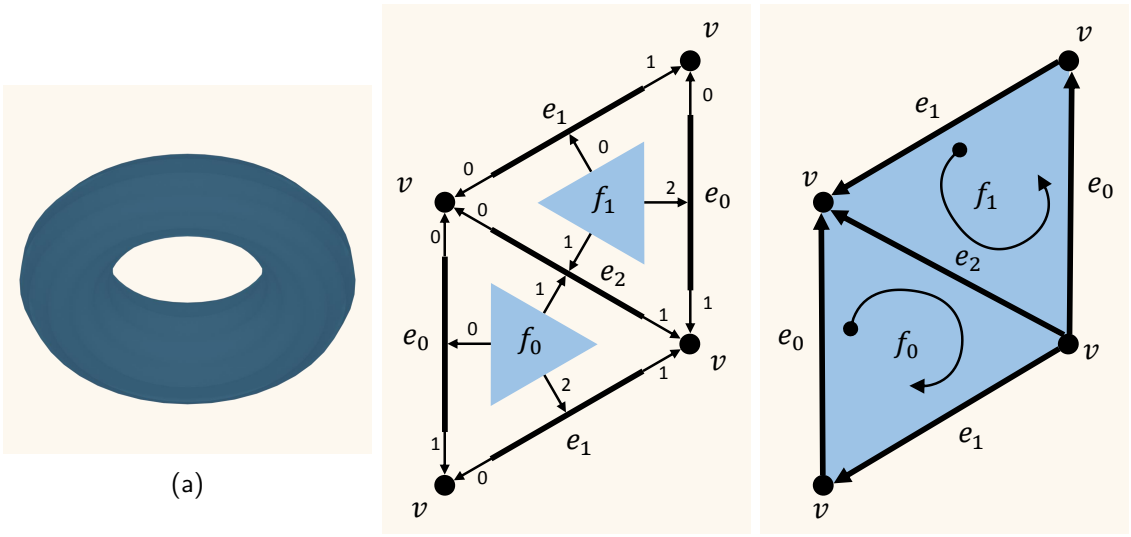
Dans cette sous-section, nous présentons quelques exemples de groupes d'homologie. Les objets considérés sont :

7. Un groupe monogène est un groupe engendré par un unique élément.

— un tore subdivisé, représenté par un ensemble semi-simplicial ;
— une sphère subdivisée, représentée par un ensemble semi-cubique ;
— une bouteille de Klein subdivisée, représentée par une carte combinatoire ;

1.3.4-a Tore - ensemble semi-simplicial

Les groupes d'homologie du tore de la figure 1.30 sont calculés sur le complexe de chaînes $(C, \partial)^2$ de la figure 1.30c induit par l'ensemble semi-simplicial de la figure 1.30b.



(a) (b) (c)

FIGURE 1.30 – Un tore. (a) Représentation géométrique. (b) L'ensemble semi-simplicial (K, d), où deux simplexes de même nom sont identifiés entre eux. (c) Le complexe de chaînes (C, ∂).

Les générateurs des cycles et des bords de (C, ∂) sont :

$$Z^0 = \{v\} \qquad B^0 = \{\}$$
$$Z^1 = \{e_0, e_1, e_2\} \qquad B^1 = \{e_1 - e_2 + e_0\}$$
$$Z^2 = \{f_0 - f_1\} \qquad B^2 = \{\}$$

Ses groupes d'homologie sont :

$$H^0 = Z^0/B^0 = \{v\}/\{\} \cong \mathbb{Z}$$
$$H^1 = Z^1/B^1 = \{e_0, e_1, e_2\}/\{e_1 - e_2 + e_0\} \cong \mathbb{Z} \oplus \mathbb{Z}$$
$$H^2 = Z^2/B^2 = \{f_0 - f_1\}/\{\} \cong \mathbb{Z}$$

Dit autrement, les nombres de Betti du tore sont $\beta^0 = 1$, $\beta^1 = 2$ et $\beta^2 = 1$.

1.3.4-b Sphère - ensemble semi-cubique

Les groupes d'homologie de la sphère de la figure 1.31 sont calculés sur le complexe de chaînes $(C, \partial)^2$ de la figure 1.31c induit par l'ensemble semi-cubique de la figure 1.31b.

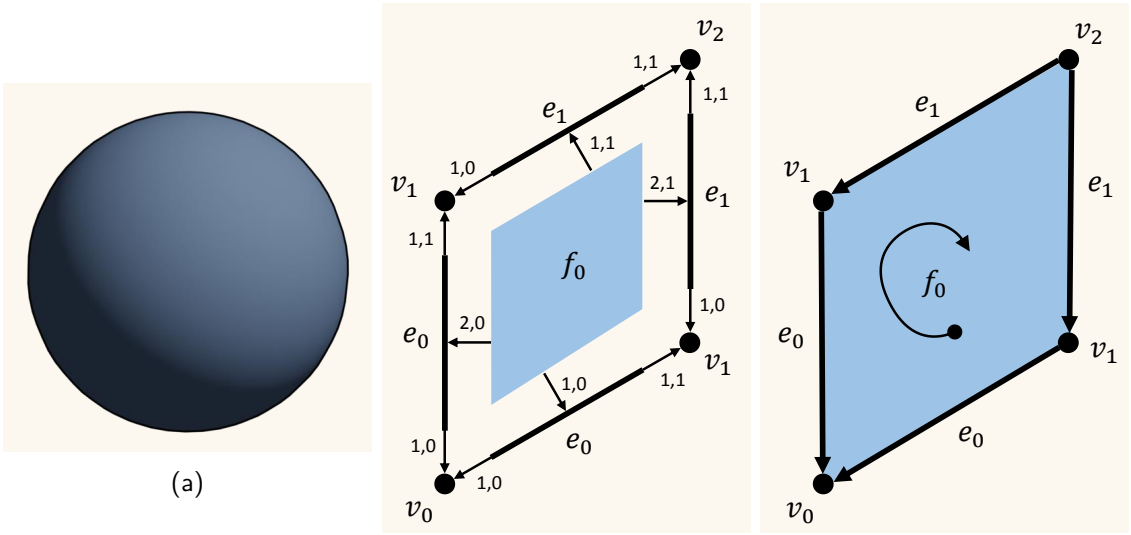


FIGURE 1.31 – Une sphère. (a) Représentation géométrique. (b) L'ensemble semi-cubique (K, d), où deux cubes de même nom sont identifiés entre eux. (c) Le complexe de chaînes (C, ∂).

Des générateurs des cycles et des bords de (C, ∂) sont :

$$Z^0 = \{v_0, v_1, v_2\} \qquad B^0 = \{v_1 - v_2, v_0 - v_1\}$$
$$Z^1 = \{\} \qquad B^1 = \{\}$$
$$Z^2 = \{f_0\} \qquad B^2 = \{\}$$

Ses groupes d'homologie sont :

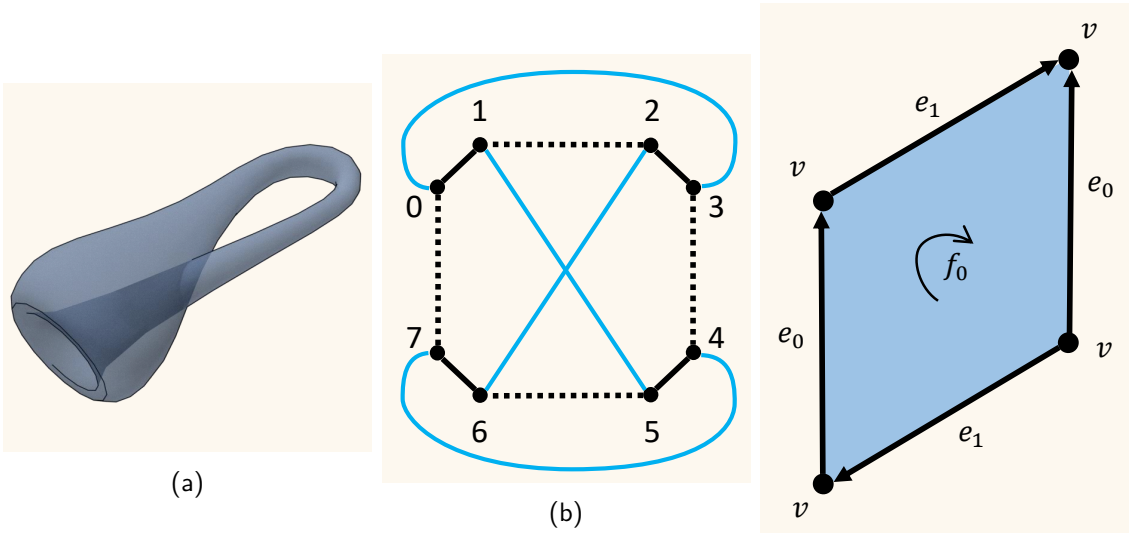
$$H^0 = Z^0/B^0 = \{v_0, v_1, v_2\}/\{v_1 - v_2, v_0 - v_1\} \cong \mathbb{Z}$$
$$H^1 = Z^1/B^1 = \{\}/\{\} \cong 0$$
$$H^2 = Z^2/B^2 = \{f_0\}/\{\} \cong \mathbb{Z}$$

Dit autrement, les nombres de Betti de la sphère sont $\beta^0 = 1$, $\beta^1 = 0$ et $\beta^2 = 1$.

1.3.4-c Bouteille de Klein - carte combinatoire

Les groupes d'homologie de la bouteille de Klein de la figure 1.32 sont calculés sur le complexe de chaînes $(C, \partial)^2$ de la figure 1.32c induit par la carte combinatoire de la figure 1.32b.
Des générateurs des cycles et des bords de (C, ∂) sont :

$$Z^0 = \{v\} \qquad B^0 = \{\}$$
$$Z^1 = \{e_0, e_1\} \qquad B^1 = \{2 * e_1\}$$
$$Z^2 = \{\} \qquad B^2 = \{\}$$



(a) (b) (c)

FIGURE 1.32 – Une bouteille de Klein. (a) Représentation géométrique. (b) La carte combinatoire (B, α). (c) Le complexe de chaînes (C, ∂).

Ses groupes d'homologie sont :

$$H^0 = Z^0/B^0 = \{v\}/\{\} \cong \mathbb{Z}$$
$$H^1 = Z^1/B^1 = \{e_0, e_1\}/\{2 * e_1\} \cong \mathbb{Z} \oplus \mathbb{Z}/2\mathbb{Z}$$
$$H^2 = Z^2/B^2 = \{\}/\{\} \cong 0$$

Dit autrement, les nombres de Betti de la bouteille de Klein sont $\beta^0 = 1$, $\beta^1 = 1$ et $\beta^2 = 0$ et 2 est un coefficient de torsion.

1.4 État de l'art

L'objectif de cette section est de positionner les travaux présentés dans cette thèse par rapport à l'existant. Pour cela, nous présentons des grandes catégories de méthodes de calcul de groupes d'homologie. Pour chacune d'elles, nous expliquons d'abord son principe en nous appuyant sur un document de référence, puis nous présentons d'autres travaux de recherche liés à cette catégorie.

1.4.1 Forme Normale de Smith

Tout morphisme peut être représenté sous forme de matrice. En particulier, c'est le cas de l'opérateur de bord ∂ des complexes de chaînes. Le calcul des groupes d'homologie au travers d'une mise en Forme Normale de Smith (FNS) repose sur cette représentation matricielle. Il est utilisable pour n'importe quel type de structure topologique dès lors que l'on sait en extraire un complexe de chaînes.

1.4.1-a Principe

Le document de référence choisi est [34]. En 1861, Henry John Stephen Smith présente dans [35] des travaux sur la résolution de systèmes d'équations indéterminées. Il y définit une forme matricielle particulière qui prendra son nom, la forme normale de Smith (FNS). En particulier, il prouve que toute matrice définie sur $\mathbb{Z}$ peut être mise en FNS.

Soit un complexe de chaînes (C, ∂). Pour tout p compris entre 1 et n inclus, une ligne de la matrice associée à $\partial^p : C^p \to C^{p-1}$, également notée ∂^p, représente le bord d'un générateur de C^p. D'après le théorème 11.3 de [34], il existe des bases de C^p et C^{p-1} telles que ∂^p soit en FNS. Un algorithme de calcul de ces bases est donné page 56 de [34].

Définition 1.4.1. *Une matrice ∂^p est en **forme normale de Smith** (FNS) si elle est de la forme :*

générateurs de C^{p-1}

générateurs de C^p	c_0^{p-1}	c_1^{p-1}	$\dots$	c_q^{p-1}	c_{q+1}^{p-1}	$\dots$	c_n^{p-1}
c_0^p	b_1	0	$\dots$	0			
c_1^p	0	$\ddots$		0		0	
$\vdots$	$\vdots$		$\ddots$	$\vdots$			
c_q^p	0	0	$\dots$	b_q			
c_{q+1}^p			0			0	
$\vdots$							
c_m^p							

où $b_j \geq 1$ et b_i divise b_{i+1} $\quad \forall q, i, j | 0 \leq q \leq m, n \quad 1 \leq j \leq q \quad 1 \leq i < q.$

L'information homologique "se lit" sur la FNS. Soit Z^p le groupe des p-cycles de (C, ∂) et B^{p-1} le groupe de ses $(p-1)$-bords :

— une ligne nulle de ∂^p correspond à un générateur c^p de Z^p car $c^p \partial^p = 0$;

— une colonne c^{p-1} non nulle de ∂^p correspond à un générateur de B^{p-1} car il existe une ligne c^p telle que $c^p \partial^p = c^{p-1}$;

(a) (b)

FIGURE 1.33 – (a) Sphère S^1. (b) Un complexe de chaînes (C, ∂) associé à une subdivision de S^1.

Le nombre de Betti β^p est la différence entre le rang de Z^p et celui de B^p. Dit autrement, il est la différence entre le nombre de lignes nulles de ∂^p et le nombre de colonnes non nulles de ∂^{p+1}. Les coefficients strictement supérieurs à 1 sont les coefficients de torsion de H^p.

Exemple. Les matrices ∂^0 et ∂^1 associées au complexe de chaînes (C, ∂) illustré sur la figure 1.33b sont :

$$\begin{array}{c} \partial^1 \\ e_0 \\ e_1 \\ e_2 \end{array} \begin{array}{c} \begin{array}{cccc} v_0 & v_1 & v_2 \end{array} \\ \begin{bmatrix} 1 & -1 & . \\ . & -1 & 1 \\ -1 & . & 1 \end{bmatrix} \end{array}$$

La matrice ∂^0 est en FNS. La FNS de ∂^1 peut être obtenue de la manière suivante :

$$\begin{array}{c} \\ e_0 \\ e_1 \\ e_2 \end{array} \begin{array}{c} \begin{array}{ccc} v_0 & v_0+v_1 & v_2 \end{array} \\ \begin{bmatrix} 1 & . & . \\ . & -1 & 1 \\ -1 & -1 & 1 \end{bmatrix} \end{array} \quad \begin{array}{c} \\ e_0 \\ -e_1 \\ e_2 \end{array} \begin{array}{c} \begin{array}{ccc} v_0 & v_0+v_1 & v_2 \end{array} \\ \begin{bmatrix} 1 & . & . \\ . & 1 & -1 \\ -1 & -1 & 1 \end{bmatrix} \end{array} \quad \begin{array}{c} \\ e_0 \\ -e_1 \\ e_2+e_0-e_1 \end{array} \begin{array}{c} \begin{array}{ccc} v_0 & v_0+v_1 & v_0+v_1+v_2 \end{array} \\ \begin{bmatrix} 1 & . & . \\ . & 1 & . \\ . & . & . \end{bmatrix} \end{array}$$

En particulier, on compte sur la FNS de ∂^1 :

— une ligne nulle, donc $rang(Z^1) = 1$;

— deux colonnes non nulles, donc $rang(B^0) = 2$.

On sait que $rang(Z^0) = rang(C^0) = 3$ car tout générateur de C^0 est un 0-cycle, et que $rang(B^1) = 0$ car C^2 est vide. D'où :

$$\beta^0 = rang(Z^0) - rang(B^0) = 1 \qquad \beta^1 = rang(Z^1) - rang(B^1) = 1$$

Enfin, les changements de base étant connus, observons que $e_2 + e_0 - e1$ est un générateur de H^1.

1.4.1-b Autour de la Forme Normale de Smith

Améliorations algorithmiques. Nous distinguons deux types d'algorithmes autour de la mise en FNS d'une matrice :

— les algorithmes probabilistes [36, 37, 38]. À notre connaissance, ces derniers ne permettent pas de calculer les générateurs d'homologie. Aussi, l'algorithme de ce type le plus récent

à ce jour semble être *l'algorithme de Las Vegas*[8] présenté dans [38]. Il se décompose en 3 parties. Soit A la matrice que l'on cherche à mettre en FNS :

1. calculer b_q de A à l'aide de l'algorithme `Largest Invariant Factor` présenté dans [40] ;
2. calculer les autres b_i de A, $1 \leq i < q$ par résolution d'une équation particulière dont l'une des composantes est une matrice J "aléatoire" ;
3. vérifier que le résultat obtenu satisfait certains critères ;

— les algorithmes déterministes [34, 41, 42, 43]. Ils permettent parfois le calcul des générateurs d'homologie. Par exemple, l'algorithme présenté dans [41] se décompose en 4 parties. Soit A la matrice que l'on cherche à mettre en FNS :

1. calculer une matrice triangulaire inférieure T ayant la même FNS que A ;
2. calculer une matrice triangulaire supérieure B ayant la même FNS que A comme étant la transposée d'une sous-matrice de T ;
3. transformer B en matrice bidiagonale supérieure C ;
4. calculer la FNS de C.

Remarque. Cet algorithme prend en entrée une matrice définie sur $\mathbb{Z}_d$; il se généralise à $\mathbb{Z}$ au prix du calcul d'un d suffisamment grand pour que la FNS de A sur $\mathbb{Z}$ soit isomorphe à celle de A sur $\mathbb{Z}_d$.

Une analyse de complexité d'une partie des algorithmes de mise en FNS est présentée dans [42, p.129]. En particulier, notons que la complexité en temps dépend systématiquement des dimensions de la matrice mise en forme.

Réduction de la donnée. En dehors des optimisations intrinsèques du calcul de la FNS, il existe des solutions consistant à réduire le complexe de chaînes avant de calculer la FNS de ∂ [44, 45]. L'idée ici est de réduire la taille de la matrice ∂^p en garantissant que l'information homologique induite par la FNS de la matrice réduite est identique à celle induite par la FNS de la matrice avant réduction.

Il existe aussi des variantes de la FNS avec lesquelles il est possible de calculer les générateurs d'homologie sans que les changements de base de toutes les matrices ∂^p ne soient explicitement conservés. Par exemple, l'algorithme de [46] garantit que la base de C^p utilisée pour les lignes de ∂^p est celle utilisée pour les colonnes de C^{p+1}.

Autres utilisations. Les propriétés de la FNS et leur interprétation sont présentées en détail dans [47]. Elle est un outil algébrique dont l'utilisation n'est pas réservée au calcul de groupes d'homologie. Elle trouve par exemple un intérêt en théorie des graphes [48], en cryptologie [49] ou en chimie [50].

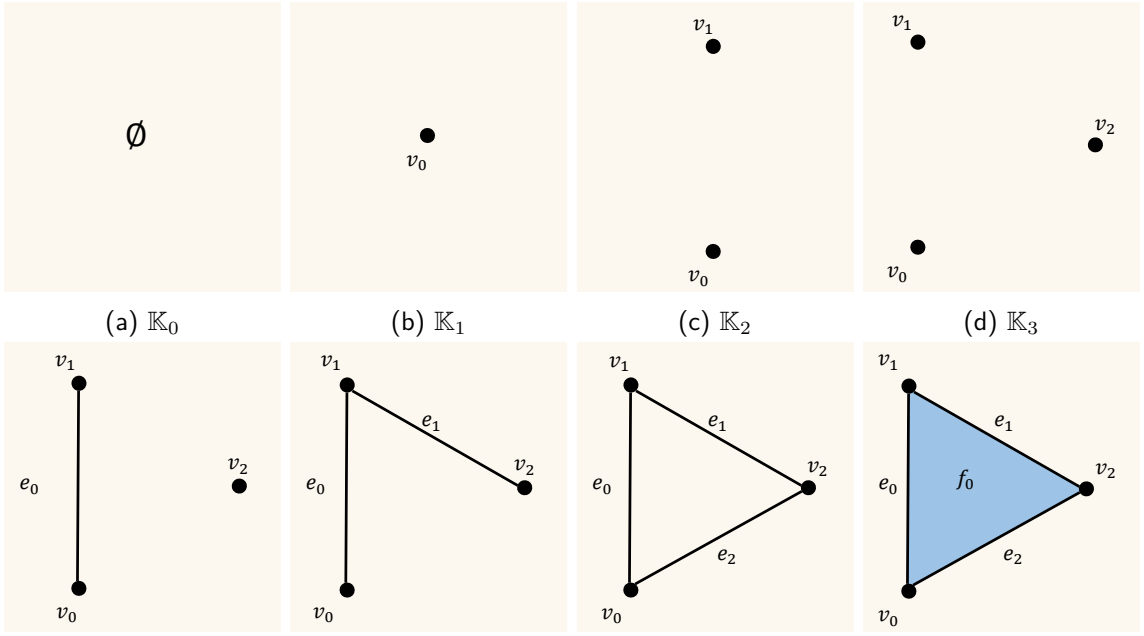
8. Algorithme probabiliste dont le résultat est toujours le bon, mais dont le temps d'exécution est aléatoire [39].

1.4.2 Homologie Persistante

L'Homologie Persistante consiste à analyser l'évolution de l'homologie d'un objet au fil de son existence, dont les **étapes** sont généralement symbolisées par une *filtration*.

1.4.2-a Principe

Le document de référence choisi est [51], et plus particulièrement le chapitre VII.



(a) $\mathbb{K}_0$ (b) $\mathbb{K}_1$ (c) $\mathbb{K}_2$ (d) $\mathbb{K}_3$

(e) $\mathbb{K}_4$ (f) $\mathbb{K}_5$ (g) $\mathbb{K}_6$ (h) $\mathbb{K}_7$

FIGURE 1.34 – Représentation graphique d'une filtration $\mathbb{K}_0 \subseteq \mathbb{K}_1 \subseteq \mathbb{K}_2 \subseteq \mathbb{K}_3 \subseteq \mathbb{K}_4 \subseteq \mathbb{K}_5 \subseteq \mathbb{K}_6 \subseteq \mathbb{K}_7 = \mathbb{K}$.

Définition 1.4.2. *Soit un complexe simplicial abstrait $\mathbb{K}$ et une application $f : \mathbb{K} \to \mathbb{R}$ vérifiant $f(\sigma) \leq f(\tau)$ quand σ est une face de τ. La **filtration** induite par f est la séquence :*

$$\emptyset = \mathbb{K}_0 \subseteq \mathbb{K}_1 \subseteq \ldots \subseteq \mathbb{K}_n = \mathbb{K}$$

où $\mathbb{K}_i = f^{-1}(-\infty, i]$ pour $0 \leq i \leq n$.

La figure 1.34 illustre une filtration.

Remarque. Dans le document de référence, les valeurs des simplexes par f sont notées a_i et peuvent être différents des indices d'étapes i. Pour simplifier, nous supposons dans ce paragraphe que $a_i = i$ pour tout i. Nous notons $H^p(\mathbb{K})$ le p-ème groupe d'homologie du complexe de chaînes associé à $\mathbb{K}$.

L'inclusion de $\mathbb{K}_i$ dans $\mathbb{K}_{i+1}$ induit une relation $f^p_{i,i+1}$ entre $H^p(\mathbb{K}_i)$ et $H^p(\mathbb{K}_{i+1})$, pour tout $0 \leq i < n$. Plus généralement, on note $f^p_{i,j} : H^p(\mathbb{K}_i) \to H^p(\mathbb{K}_j)$ la relation associant $H^p(\mathbb{K}_i)$ à $H^p(\mathbb{K}_j)$ pour $0 \leq i \leq j \leq n$. On a :

$$0 = H^p(\mathbb{K}_0) \xrightarrow{f^p_{0,1}} H^p(\mathbb{K}_1) \xrightarrow{f^p_{1,2}} \ldots \xrightarrow{f^p_{n-1,n}} H^p(\mathbb{K}_n) = H^p(\mathbb{K})$$

Un groupe d'homologie persistante, noté $H^p_{i,j}$, représente l'ensemble des classes d'homologie de l'étape i ayant survécu jusqu'à l'étape j. En particulier, $H^p_{i,i} = H^p(\mathbb{K}_i)$.

Définition 1.4.3. *Les $H^p_{i,j} = H^p_i f^p_{i,j}$ sont les p-**groupes d'homologie persistante**, $\forall i, j | 0 \leq i \leq j \leq n$.*

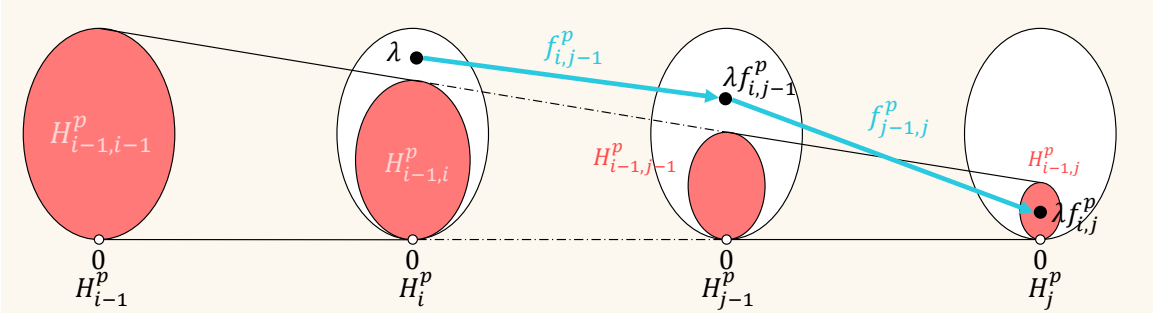


FIGURE 1.35 – Évolution d'un groupe d'homologie H^p_{i-1} dans une filtration. En rose, la persistance de H^p_{i-1} à différentes étapes. En bleu, les relations $f^p_{i,j-1}$ et $f^p_{j-1,j}$. λ est une classe d'homologie née en i et morte en j. Cette figure est inspirée de [51, p.180]

Le **nombre de Betti persistant** $\beta^p_{i,j}$ est le rang de $H^p_{i,j}$ Une classe d'homologie λ de $H^p(\mathbb{K}_i)$ **née** en i si elle n'appartient pas à $H^p_{i-1,i}$, et **meurt** en j si :

— $\lambda f^p_{i,j-1}$ n'appartient pas à $H^p_{i-1,j-1}$;
— et $\lambda f^p_{i,j}$ appartient à $H^p_{i-1,j}$.

La figure 1.35 illustre la naissance et la mort d'une classe d'homologie λ. Le nombre de p-classes d'homologie nées en i et mortes en j est :

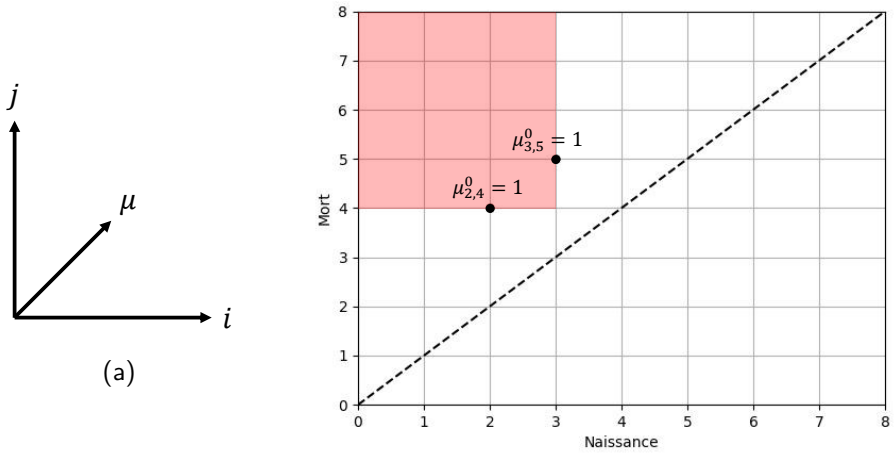
$$\mu^p_{i,j} = (\beta^p_{i,j-1} - \beta^p_{i,j}) - (\beta^p_{i-1,j-1} - \beta^p_{i-1,j})$$

Diagramme de persistance. L'**indice de persistance** d'une classe d'homologie λ, noté $pers(\lambda)$, est la différence entre son indice de mort j et son indice de naissance i. Les indices de persistance sont généralement représentés sous la forme d'un **diagramme de persistance**. Le diagramme de persistance de dimension p est composé de l'ensemble des points $(i, j, \mu^p_{i,j})$. Ce dernier est représentable en deux dimensions (i, j) à condition que chaque point soit annoté de $\mu^p_{i,j}$. Pour déterminer $\beta^p_{i,j}$, on compte le nombre de points (k, l) tels que $k \leq i$ et $l > j$. Intuitivement, cela revient à compter le nombre de points du diagramme dans le quadrant haut-gauche de coin (k, l), fermé à droite et ouvert en bas.

Propriété 1.2. Le diagramme de persistance encode la totalité de l'information liée à l'homologie persistante d'une filtration [51, p.181].

Remarque. Une classe d'homologie qui ne meurt pas a un indice de persistance à l'infini, et n'est pas représentée sur le diagramme de persistance.

Calcul des indices de persistance. Les indices de persistance sont calculés à partir d'une matrice M induite par $f : \mathbb{K} \to \mathbb{R}$. Les lignes et colonnes de M représentent les m simplexes de $\mathbb{K}$ dans l'ordre $\sigma_0, \ldots, \sigma_{m-1}$ vérifiant : pour tous indices x, y compris entre 0 et $m-1$, $x < y$ si,



(a) (b)

FIGURE 1.36 – (a) Les 3 dimensions d'un diagramme de persistance. (b) Une représentation en deux dimensions du diagramme de persistance de dimension 0 de la filtration illustrée sur la figure 1.34. Le point $\mu^0_{2,4} = 1$ représente la classe d'équivalence créée par v_1 dans $\mathbb{K}_1$ et tuée par e_0 dans $\mathbb{K}_4$. Le point $\mu^0_{3,5} = 1$ représente la classe d'équivalence créée par v_2 dans $\mathbb{K}_2$ et tuée par e_1 dans $\mathbb{K}_5$. En rose, le quadrant utilisé pour déterminer $\beta^0_{3,4} = 1 : \mu^0_{3,5}$ compte et $\mu^0_{2,4}$ ne compte pas.

— $f(\sigma_x) \leq f(\sigma_y)$;
— σ_x est une face de σ_y.

Le coefficient $M[x, y]$ de la ligne x et colonne y de M est :

$$M[x,y] = \begin{cases} 1 \text{ si le } p\text{-simplexe } \sigma_x \text{ est une face du } p'\text{-simplexe } \sigma_y, \text{et que } p' - p = 1 \\ 0 \text{ sinon} \end{cases}$$

Notons $low(y) = x$ le fait que x soit la plus basse ligne de la colonne y où $M[x, y] \neq 0$. L'algorithme ci-dessous modifie la matrice M en la balayant de gauche à droite, et en faisant en sorte qu'à la fin de son exécution il n'existe pas deux colonnes y et y_0 telles que $low(y) = low(y')$. La matrice ainsi obtenue est dite **réduite**, et est notée R_M.

Algorithme : Réduction de M

Entrées : Matrice $M_{m \times m}$

```
pour y de 1 à m faire
    tant que ∃y0|y0 < y, low(y0) = low(y) faire
        Ajouter la colonne y0 à la colonne y dans M
    fin
fin
```

Pour chaque colonne non nulle y de R_M telle que $\sigma_{low(y)}$ est un p-simplexe, le couple $(low(y), y)$ représente un point du diagramme de persistance de dimension p. En particulier :

— le nombre de colonnes nulles représentant un p-simplexe est le rang de Z^p, et est notée $\#Zero_p(R_M)$;

— le nombre de lignes $low(y)$ représentant un p-simplexe est le rang de B^p, et est notée $\#Low_p(R_M)$.

Il en découle $\beta^p = \#Zero_p(R_M) - \#Low_p(R_M)$.

Remarque. La matrice réduite n'est pas unique mais l'ensemble des couples $(low(y), y)$ l'est. Notons que, comme pour l'utilisation de la FNS, il convient de conserver les changements de base effectués au travers des ajouts de colonnes pour obtenir les générateurs d'homologie. De plus, les coefficients de M et R_M sont nécessairement dans $\mathbb{Z}_2$.

Exemple. Le complexe simplicial $\mathbb{K}$ illustré sur la figure 1.34h est défini par :

$$\{v_0, v_1, v_2, e_0, e_1, e_2, f_0\}$$

L'application $f : \mathbb{K} \to \mathbb{R}$ de la filtration est :

— $f(v_0) = 1$, $f(v_1) = 2$, $f(v_2) = 3$;

— $f(e_0) = 4$, $f(e_1) = 5$, $f(e_2) = 6$;

— $f(f_0) = 7$.

La matrice M induite par $f : \mathbb{K} \to \mathbb{R}$ et sa réduction R sont :

$$\begin{array}{c|ccccccc} M & v_0 & v_1 & v_2 & e_0 & e_1 & e_2 & f_0 \\ v_0 & . & . & . & 1 & . & 1 & . \\ v_1 & . & . & . & 1 & 1 & . & . \\ v_2 & . & . & . & . & 1 & 1 & . \\ e_0 & . & . & . & . & . & . & 1 \\ e_1 & . & . & . & . & . & . & 1 \\ e_2 & . & . & . & . & . & . & 1 \\ f_0 & . & . & . & . & . & . & . \end{array} \qquad \begin{array}{c|ccccccc} R_M & v_0 & v_1 & v_2 & e_0 & e_1 & e_2+e_1+e_0 & f_0 \\ v_0 & . & . & . & 1 & . & . & . \\ v_1 & . & . & . & 1 & 1 & . & . \\ v_2 & . & . & . & . & 1 & . & . \\ e_0 & . & . & . & . & . & . & 1 \\ e_1 & . & . & . & . & . & . & 1 \\ e_2 & . & . & . & . & . & . & 1 \\ f_0 & . & . & . & . & . & . & . \end{array}$$

Les différents états de M au cours de l'algorithme de réduction sont :

$$\begin{array}{c|ccccccc} M & v_0 & v_1 & v_2 & e_0 & e_1 & e_2 & f_0 \\ v_0 & . & . & . & 1 & . & 1 & . \\ v_1 & . & . & . & 1 & 1 & . & . \\ v_2 & . & . & . & . & 1 & 1 & . \\ e_0 & . & . & . & . & . & . & 1 \\ e_1 & . & . & . & . & . & . & 1 \\ e_2 & . & . & . & . & . & . & 1 \\ f_0 & . & . & . & . & . & . & . \end{array} \qquad \begin{array}{c|ccccccc} & v_0 & v_1 & v_2 & e_0 & e_1 & e_2+e_1 & f_0 \\ v_0 & . & . & . & 1 & . & 1 & . \\ v_1 & . & . & . & 1 & 1 & 1 & . \\ v_2 & . & . & . & . & 1 & . & . \\ e_0 & . & . & . & . & . & . & 1 \\ e_1 & . & . & . & . & . & . & 1 \\ e_2 & . & . & . & . & . & . & 1 \\ f_0 & . & . & . & . & . & . & . \end{array}$$

$$\begin{array}{c|ccccccc} R_M & v_0 & v_1 & v_2 & e_0 & e_1 & e_2+e_1+e_0 & f_0 \\ v_0 & . & . & . & 1 & . & . & . \\ v_1 & . & . & . & 1 & 1 & . & . \\ v_2 & . & . & . & . & 1 & . & . \\ e_0 & . & . & . & . & . & . & 1 \\ e_1 & . & . & . & . & . & . & 1 \\ e_2 & . & . & . & . & . & . & 1 \\ f_0 & . & . & . & . & . & . & . \end{array}$$

En particulier, les indices de persistance se déduisent de R_M :

— $(2, 4)$, déduit de la colonne e_0, est un point du diagramme de persistance de dimension 0 car la ligne 2 est v_2 ;

— $(3, 5)$, déduit de la colonne e_1, est un point du diagramme de persistance de dimension 0 ;

— $(6, 7)$, déduit de la colonne f_0, est un point du diagramme de persistance de dimension 1 ;

D'où le diagramme de la figure 1.36.

1.4.2-b Autour de l'Homologie Persistante

Extension de son contexte. Les travaux de recherche présentés dans ce paragraphe sont des extensions du contexte applicatif de l'Homologie Persistante. Les limites ainsi repoussées sont par exemple :

— la nécessité d'avoir une inclusion de $\mathbb{K}_i$ dans $\mathbb{K}_{i+1}$. Les pistes explorées sont, entre autres,

  · la **persistance zigzag** [52, 53, 54], avec laquelle le "sens" des relations d'inclusion de la filtration n'importe plus ($\mathbb{K}_i \supseteq \mathbb{K}_{i+1}$ est autorisé) ;

  · la **persistance de tours** [2, 19], où les complexes simpliciaux ne sont plus reliés par une inclusion mais, plus généralement, par une application simpliciale ;

— l'étude d'une seule fonction de filtration à la fois. L'extension, appelée **persistance multidimensionnelle** (ou persistance multiparamètres) [55, 56, 57] consiste à étudier la persistance des groupes d'homologie d'un objet au travers de plusieurs fonctions.

— l'absence de prise en compte de la torsion. Les pistes explorées sont :

  · le calcul sur d'autres corps que $\mathbb{Z}_2$ et la comparaison des résultats obtenus [58]. L'idée est que la totalité de l'information homologique est caractérisée lorsque les résultats convergent ;

  · l'inférence du corps à utiliser [59]. Intuitivement, on cherche à déterminer à l'avance les coefficients à utiliser pour garantir que la totalité de l'information homologique est caractérisée ;

Remarque. À notre connaissance, et comme indiqué en partie 5 de [58], l'homologie persistante n'est pas utilisable avec des coefficients dans $\mathbb{Z}$.

Améliorations algorithmiques. Les travaux de recherche présentés dans ce paragraphe sont des améliorations du principe de base de l'homologie persistante tel que présentée dans le paragraphe 1.4.2-a. Nous distinguons :

— l'utilisation d'une étape de pré-calcul consistant à réduire la taille du complexe simplicial filtré : [60, 61, 62] pour le cas général, [54] pour la persistance zigzag et [56] pour la persistance multi dimensionnelle ;

— l'exploitation d'un nuage de point associé au complexe simplicial pour approximer sa filtration [63, 64, 65, 66, 67]. Généralement, l'idée est de regrouper certains points selon des critères bien définis. Par exemple, dans [66], on suppose que les points appartiennent à un espace métrique, et la distance entre les points est utilisée pour les regrouper.

Analyse topologique de donnée. À ce jour, l'homologie persistante est, selon nous, la catégorie la plus travaillée en recherche. Ceci s'explique en partie par l'émergence de l'analyse topologique de données (TDA) [68, 69]. Parmi les cas applicatifs de la TDA, on retrouve :

— la médecine [70, 5, 31, 71] ;

— la biologie [72, 73] ;

— le domaine pharmaceutique [74].

Dans ces exemples, l'objet manipulé est un nuage de points à partir duquel un complexe simplicial est reconstruit. Les reconstructions les plus classiques sont les reconstructions Alpha [75], Čech [76] et Vietoris-Rips [77].

1.4.3 Homologie effective

Le document de référence choisi est [16], basé sur les travaux de [1]. L'Homologie Effective, au sens de [1], est une sous-classe *constructive* de la topologie algébrique, c'est-à-dire adaptée à la conception d'algorithmes.

1.4.3-a Principe

Son principe repose sur deux notions :

— *l'équivalence homologique*, dont le rôle est d'associer un complexe de chaînes à un plus petit [9] complexe de chaînes de même homologie et sur lequel l'homologie est calculée [10] (avec une mise en forme normale de Smith par exemple) ;

— le *théorème des suites exactes courtes effectives*, à l'aide duquel on sait construire l'équivalence homologique de l'étape $t+1$ étant donnée une *suite exacte courte effective* symbolisant le passage de t à $t+1$.

Equivalence homologique.

Définition 1.4.4. *Soient deux complexes de chaînes (C, ∂) et (C', ∂'). Une **réduction** ρ de (C, ∂) sur (C', ∂') est composée de deux morphismes de complexe de chaînes $f : C \to C'$ et $g : C' \to C$, et d'un morphisme gradué $h : C \to C$ de degré 1, qui vérifient :*

— $gf = id_{C'}$;

— $fg + h\partial + \partial h = id_C$;

— $hf = 0_{C'}$, $gh = 0_C$ *et* $hh = 0_C$.

Remarque. La réduction ρ est notée $\rho : (C, \partial) \Rrightarrow (C', \partial')$ par la suite. Une **réduction identité** est telle que f est un morphisme identité, g est un morphisme identité, et h est un morphisme nul.

Propriété 1.3. S'il existe une réduction entre deux complexes de chaînes (C, ∂) et (C', ∂'), alors les groupes d'homologie de (C, ∂) et (C', ∂') sont isomorphes.

9. C'est-à-dire composé de moins de générateurs.
10. Le calcul est donc plus efficace que s'il était fait sur le "plus gros" complexe de chaînes.

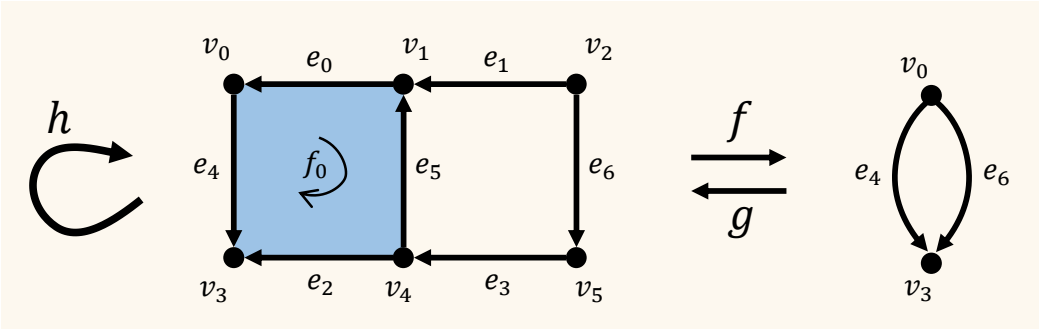


FIGURE 1.37 – Représentation graphique d'une réduction $\rho : (C, \partial) \Rrightarrow (C', \partial')$ où (C, ∂) est à gauche et (C', ∂') est à droite.

Exemple. La réduction illustrée sur la figure 1.37 est définie par :

$f : C \to C'$	$g : C' \to C$	$h : C \to C$
$v_0 f = v_0$	$v_0 g = v_0$	$v_0 h = 0$
$v_1 f = v_0$		$v_1 h = -e_0$
$v_2 f = v_0$		$v_2 h = -e_1 - e_0$
$v_3 f = v_3$	$v_3 g = v_3$	$v_3 h = 0$
$v_4 f = v_3$		$v_4 h = -e_2$
$v_5 f = v_3$		$v_5 h = -e_3 - e_2$
$e_0 f = 0$		$e_0 h = 0$
$e_1 f = 0$		$e_1 h = 0$
$e_2 f = 0$		$e_2 h = 0$
$e_3 f = 0$		$e_3 h = 0$
$e_4 f = e_4$	$e_4 g = e_4$	$e_4 h = 0$
$e_5 f = -e_4$		$e_5 h = -f_0$
$e_6 f = e_6$	$e_6 g = e_6 + e_3 + e_2 - e_1 - e_0$	$e_6 h = 0$
$f_0 f = 0$		

En particulier :

- $e_6 g f = e_6$
- $e_6 f g + e_6 h \partial + e_6 \partial h = (e_6 + e_3 + e_2 - e_1 - e_0) + 0 + ((-e_3 - e_2) - (-e_1 - e_0)) = e_6$
- $e_6 h f = 0$, $e_6 g h = 0$ et $e_6 h h = 0$

Définition 1.4.5. *Soient les complexes de chaînes (C, ∂), (C^B, ∂^B), (C^S, ∂^S). Une* ***équivalence homologique*** *γ est composée de deux réductions ρ et ρ^S telle que :*

$$\gamma : (C, \partial) \overset{\rho}{\Lleftarrow} (C^B, \partial^B) \overset{\rho^S}{\Longrightarrow} (C^S, \partial^S)$$

Remarque. Les groupes d'homologie de (C, ∂), (C^B, ∂^B) et (C^S, ∂^S) d'une équivalence homologique sont isomorphes d'après la propriété 1.3.

Exemple. La figure 1.38 illustre une équivalence homologique $\gamma : (C, \partial) \overset{\rho}{\Lleftarrow} (C^B, \partial^B) \overset{\rho^S}{\Longrightarrow} (C^S, \partial^S)$. La réduction ρ est caractérisée par [11] :

11. ρ et ρ^S sont des *réductions élémentaires*. Cette notion est définie en annexe A.

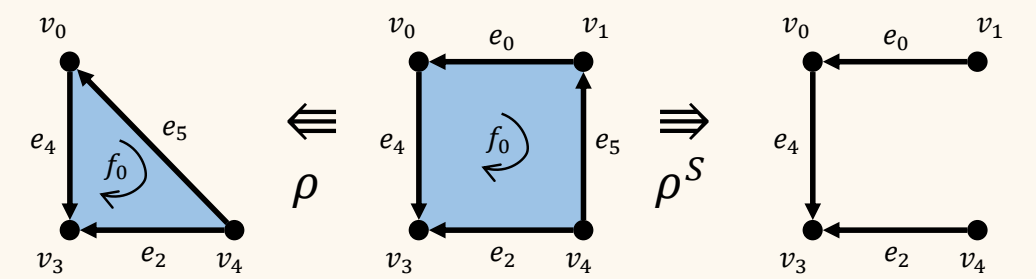


FIGURE 1.38 – Représentation graphique d'une équivalence homologique $(C, \partial) \overset{\rho}{\Longleftarrow} (C^B, \partial^B) \overset{\rho^S}{\Longrightarrow} (C^S, \partial^S)$.

— $v_1 h = -e_0$, et h est nul partout ailleurs ;
— $v_1 f = v_0$, $e_0 = 0$ et f est l'identité partout ailleurs ;
— $e_5 g = e_5 + e_0$ et g est l'identité partout ailleurs.

La réduction ρ^S est caractérisée par :

— $e_5 h^S = -f_0$ et h est nul partout ailleurs ;
— $e_5 f^S = e_2 - e_4 - e_0$, $f_0 f = 0$ et f est l'identité partout ailleurs ;
— g^S est l'inclusion de C^S dans C^B.

En particulier, tous les complexes de chaînes représentés ont les mêmes nombres de Betti, à savoir $\beta^0 = 1$, $\beta^1 = 0$ et $\beta^2 = 0$.

Théorème des suites exactes courtes effectives.

Définition 1.4.6. *Soient les complexes de chaînes (C, ∂), (C', ∂') et (C'', ∂''). Une **suite exacte courte effective** est composée :*

— *des morphismes de complexes de chaînes $i : (C, \partial) \to (C', \partial')$ et $j : (C', \partial') \to (C'', \partial'')$,*
— *des morphismes gradués $s : (C'', \partial'') \to (C', \partial')$ et $r : (C', \partial') \to (C, \partial)$,*

qui vérifient :

1. $ir = id_C$
2. $ri + js = id_{C'}$
3. $sj = id_{C''}$

Remarque. Par la suite, une suite exacte courte effective (SECE) est représentée par :

$$(C, \partial) \underset{\mathsf{r}}{\overset{\mathsf{i}}{\rightleftarrows}} (C', \partial') \underset{\mathsf{s}}{\overset{\mathsf{j}}{\rightleftarrows}} (C'', \partial'')$$

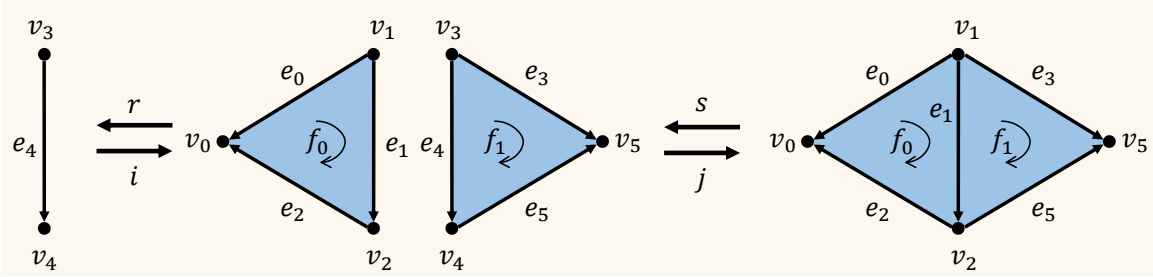


FIGURE 1.39 – Représentation graphique d'une suite exacte courte effective. À gauche, le complexe de chaînes (C, ∂). Au milieu, le complexe de chaînes (C', ∂'). À droite, le complexe de chaînes (C'', ∂'').

Exemple. La SECE illustrée sur la figure 1.39 est définie par :

$r : C' \to C$	$i : C \to C'$	$j : C' \to C''$	$s : C'' \to C'$
$v_0 r = 0$		$v_0 j = v_0$	$v_0 s = v_0$
$v_1 r = 0$		$v_1 j = v_1$	$v_1 s = v_1$
$v_2 r = 0$		$v_2 j = v_2$	$v_2 s = v_2$
$v_3 r = -v_3$	$v_3 i = v_1 - v_3$	$v_3 j = v_1$	
$v_4 r = -v_4$	$v_4 i = v_2 - v_4$	$v_4 j = v_2$	
$v_5 r = 0$		$v_5 j = v_5$	$v_5 s = v_5$
$e_0 r = 0$		$e_0 j = e_0$	$e_0 s = e_0$
$e_1 r = 0$		$e_1 j = e_1$	$e_1 s = e_1$
$e_2 r = 0$		$e_2 j = e_2$	$e_2 s = e_2$
$e_3 r = 0$		$e_3 j = e_3$	$e_3 s = e_3$
$e_4 r = -e_4$	$e_4 i = e_1 - e_4$	$e_4 j = e_1$	
$e_5 r = 0$		$e_5 j = e_5$	$e_5 s = e_5$
$f_0 r = 0$		$f_0 j = f_0$	$f_0 s = f_0$
$f_1 r = 0$		$f_1 j = f_1$	$f_1 s = f_1$

En particulier :

- $e_4 ir = e_4$
- $e_4 ri + e_4 js = (-e_1 + e_4) + (e_1) = e_4$
- $e_1 sj = e_1$

La figure 1.40 illustre une partie des éléments utilisés dans le théorème présenté ci-dessous.

Théorème 2. *Soit une suite exacte courte effective :*

$$(C, \partial) \underset{r}{\overset{i}{\rightleftarrows}} (C', \partial') \underset{s}{\overset{j}{\rightleftarrows}} (C'', \partial'')$$

Nous distinguons deux cas :

1. *Étant données deux équivalences homologiques :*
   - $\gamma : (C, \partial) \overset{\rho}{\Longleftarrow} (C^B, \partial^B) \overset{\rho^S}{\Longrightarrow} (C^S, \partial^S)$ *et ;*
   - $\gamma' : (C', \partial') \overset{\rho'}{\Longleftarrow} (C'^B, \partial'^B) \overset{\rho'^S}{\Longrightarrow} (C'^S, \partial'^S)$

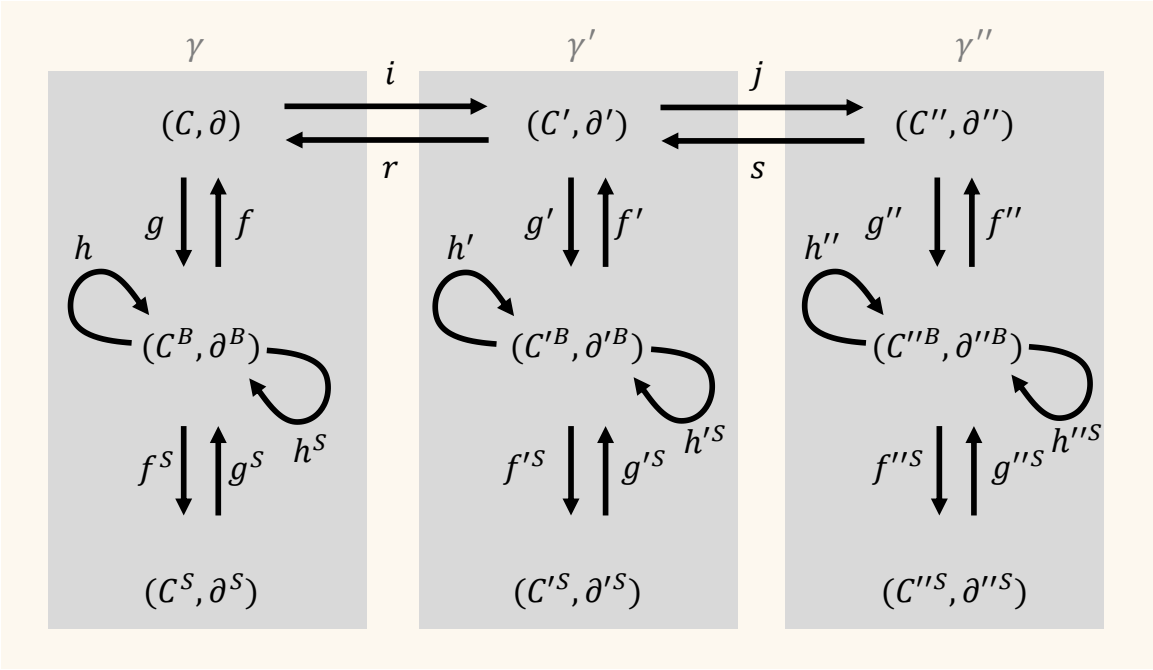


FIGURE 1.40 – Représentation d'éléments impliqués dans le théorème des suites exactes courtes effectives. Les équivalences homologiques sont mises en évidence via un fond gris.

Il est possible de calculer une équivalence homologique :

$$\gamma'' : (C'', \partial'') \xleftarrow{\rho''} (C''^B, \partial''^B) \xRightarrow{\rho''^S} (C''^S, \partial''^S)$$

.

2. *Étant données deux équivalences homologiques :*
   — $\gamma : (C, \partial) \Leftarrow (C^B, \partial^B) \Rightarrow (C^S, \partial^S)$ *et ;*
   — $\gamma'' : (C'', \partial'') \Leftarrow (C''^B, \partial''^B) \Rightarrow (C''^S, \partial''^S)$.

   Il est possible de calculer une équivalence homologique :

$$\gamma' : (C', \partial') \xleftarrow{\rho'} (C'^B, \partial'^B) \xRightarrow{\rho'^S} (C'^S, \partial'^S)$$

Considérons le cas 1 du théorème. Pour calculer l'équivalence homologique γ'', définissons les morphismes de complexe de chaînes $i^B : C^B \to C'^B$ et $i^S : C^S \to C'^S$ par :

$$i^B = fig' \qquad i^S = g^S i^B f'^S$$

L'équivalence homologique $\gamma'' : (C'', \partial'') \xleftarrow{\rho''} (C''^B, \partial''^B) \xRightarrow{\rho''^S} (C''^S, \partial''^S)$ est alors définie par :

$$C''^{B,p} = C^{B,p-1} \bigoplus C'^{B,p} \qquad \partial''^{B,p} = \begin{pmatrix} -\partial^{B,p-1} & i^{B,p-1} \\ 0 & \partial'^{B,p} \end{pmatrix}$$

$$C''^{S,p} = C^{S,p-1} \bigoplus C'^{S,p} \qquad \partial''^{S,p} = \begin{pmatrix} -\partial^{S,p-1} & i^{S,p-1} \\ 0 & \partial'^{S,p} \end{pmatrix}$$

$$f'' = \begin{pmatrix} 0 \\ f'j \end{pmatrix} \qquad g'' = \begin{pmatrix} -s\partial' rg & sg' \end{pmatrix} \qquad h'' = \begin{pmatrix} -h & 0 \\ f'rg & h' \end{pmatrix}$$

$$f''^S = \begin{pmatrix} f^S & h^S i^B f'^S \\ 0 & f'^S \end{pmatrix} g''^S = \begin{pmatrix} g^S & -g^S i^B h'^S \\ 0 & g'^S \end{pmatrix} h''^S = \begin{pmatrix} -h^S & h^S i^B h'^S \\ 0 & h'^S \end{pmatrix}$$

Considérons le cas 2 du théorème. Pour calculer l'équivalence homologique γ', définissons les morphismes de complexes de chaînes $\chi : C'' \to C$, $\chi^B : C''^B \to C^B$ et $\chi^S : C''^S \to C^S$ par :

$$\chi = s\partial' r \qquad \chi^B = f''\chi g \qquad \chi^S = g''^S \chi^B f^S$$

L'équivalence homologique $\gamma' : (C', \partial') \overset{\rho'}{\Longleftarrow} (C'^B, \partial'^B) \overset{\rho'^S}{\Longrightarrow} (C'^S, \partial'^S)$ est alors définie par :

$$C'^{B,p} = C^{B,p} \bigoplus C''^{B,p} \qquad \partial'^{B,p} = \begin{pmatrix} \partial''^{B,p} & \chi^{B,p} \\ 0 & \partial^{B,p} \end{pmatrix}$$

$$C'^{S,p} = C^{S,p} \bigoplus C''^{S,p} \qquad \partial'^{S,p} = \begin{pmatrix} \partial''^{S,p} & \chi^{S,p} \\ 0 & \partial^{S,p} \end{pmatrix}$$

$$f' = \begin{pmatrix} f''s \\ fi \end{pmatrix} \qquad g' = \begin{pmatrix} jg'' & rg \end{pmatrix} \qquad h' = \begin{pmatrix} h'' & 0 \\ 0 & h \end{pmatrix}$$

$$f'^S = \begin{pmatrix} f''^S & -h''^S \chi^B f^S \\ 0 & f^S \end{pmatrix} g'^S = \begin{pmatrix} g''^S & -g''^S \chi^B h^S \\ 0 & g^S \end{pmatrix} h'^S = \begin{pmatrix} h''^S & -h''^S \chi^B h^S \\ 0 & h^S \end{pmatrix}$$

Remarque. Nous présentons ici un extrait du théorème SECE. Il est présenté en intégralité dans [1, p.71]. En particulier, nous omettons le cas dont nous n'avons pas besoin par la suite, où, connaissant γ' et γ'', on calcule γ.

1.4.3-b Autour de l'Homologie Effective

Suites exactes de Mayer-Vietoris. Une utilisation de l'homologie effective est présentée dans le rapport de recherche [78]. Ces travaux sont appliqués dans [79]. Intuitivement, l'objectif est de calculer l'homologie d'une "forme" décomposée [12] en plusieurs composantes. L'homologie de la "forme" est calculée itérativement, à partir de l'homologie des composantes, de celle de leur union, et de celle de leur intersection. Ce calcul repose en particulier sur l'utilisation de suites exactes courtes effectives de Mayer-Vietoris.

Définition 1.4.7. *Soient les complexes de chaînes A, B et $X = A \cup B$. Une **suite exacte courte effective de Mayer-Vietoris** est une suite exacte courte effective*

$$A \cap B \underset{r}{\overset{i}{\rightleftarrows}} A \oplus B \underset{s}{\overset{j}{\rightleftarrows}} X = A \cup B$$

telle que les morphismes i, j, r et s sont définis par :

12. La décomposition utilisée est bien particulière, nommée *Manifold-connected decomposition*. L'une de ses particularités est de minimiser l'intersection entre composantes.

— $\sigma i = (\sigma, \sigma)$
— $(\sigma, \sigma')j = \sigma - \sigma'$
— $(\sigma, \sigma')r = \sigma'|_{A\cap B}$
— $\sigma s = (\sigma|_A, -\sigma|_B + \sigma|_{A\cap B})$

Remarque. La notion de sous-complexe simplicial est définie dans [78] page 18. La construction du complexe de chaînes associé à un sous-complexe simplicial est explicitée à la page 23.

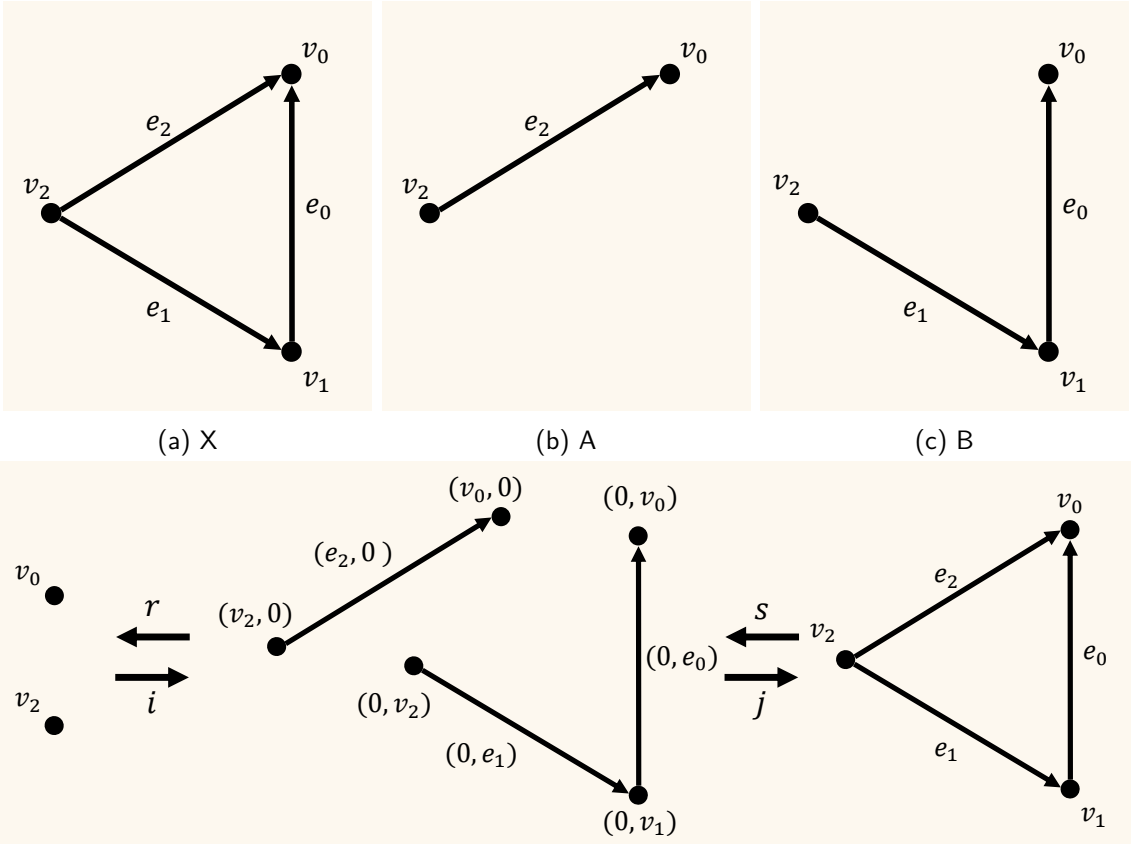


FIGURE 1.41 – Représentation graphique de trois complexes de chaînes $X = A \cup B$, A et B et une suite exacte courte effective de Mayer-Vietoris.

Exemple. La suite exacte courte effective de Mayer-Vietoris illustrée sur la figure 1.41 est définie

par :

$r : (A \oplus B) \to (A \cap B)$	$i : (A \cap B) \to (A \oplus B)$	$j : (A \oplus B) \to X$	$s : X \to (A \oplus B)$
$(v_0, 0)r = 0$	$v_0 i = (v_0, v_0)$	$(v_0, 0)j = v_0$	$v_0 s = (v_0, 0)$
$(v_0, v_0)r = v_0$		$(v_0, v_0)j = 0$	
$(0, v_0)r = v_0$		$(0, v_0)j = -v_0$	$-v_0 s = (-v_0, 0)$
$(v_2, 0)r = 0$	$v_2 i = (v_2, v_2)$	$(v_2, 0)j = v_2$	$v_2 s = (v_2, 0)$
$(0, v_2)r = v_2$		$(0, v_2)j = -v_2$	$-v_2 s = (-v_2, 0)$
$(0, v_1)r = v_1$		$(0, v_1)j = -v_1$	$-v_1 s = (0, v_1)$
$(e_2, 0)r = 0$		$(e_2, 0)j = e_2$	$e_2 s = (e_2, 0)$
$(0, e_1)r = 0$		$(0, e_1)j = -e_1$	$-e_1 s = (0, e_1)$
$(0, e_0)r = 0$		$(0, e_0)j = -e_0$	$e_0 s = (0, e_0)$
$\dots$	$\dots$	$\dots$	$\dots$

En particulier :

— $v_0 ir = (v_0, v_0)r = v_0$;

— $(v_0, 0)ri + (v_0, 0)js = 0 + (v_0, 0) = (v_0, 0)$;

— $-v_2 sj = (-v_2, 0)j = -v_2$.

1.4.4 Positionnement

Dans cette thèse, nous avons choisi d'explorer la piste de l'homologie effective. Pour rappel, notre cadre d'étude,

— implique un processus de construction : l'objet subdivisé analysé évolue au cours de différentes étapes ;

— suppose que le passage de l'étape t à $t + 1$ se fait par application d'une identification ou d'une désidentification.

Dans les paragraphes suivants, nous motivons ce choix et positionnons les grandes catégories de méthodes de calcul de groupes d'homologie présentées dans l'état de l'art par rapport à notre cadre d'étude.

1.4.4-a Forme Normale de Smith (FNS)

Rappelons que la donnée initiale du calcul des groupes d'homologie par mise en Forme Normale de Smith (FNS) est l'ensemble des matrices ∂^p induites par le complexe de chaînes (C, ∂) auquel on s'intéresse. La FNS est utilisable dans notre cadre d'étude. Cependant, nous constatons que :

— l'homologie de l'étape $t + 1$ est calculée sans tenir compte de l'homologie de l'étape t ;

— la complexité du calcul est fonction du nombre de générateurs de C à chaque étape.

Nous avons choisi de ne pas utiliser cette catégorie du fait de l'absence de prise en compte de l'évolution de l'objet subdivisé.

1.4.4-b Homologie persistante

Rappelons que, dans son principe de base, l'Homologie Persistante repose sur la notion de filtration. Nous constatons que :

— l'évolution de l'objet subdivisé est prise en compte. Elle est représentée par la filtration, et nécessite donc qu'il existe une relation d'inclusion entre les étapes t et $t+1$;

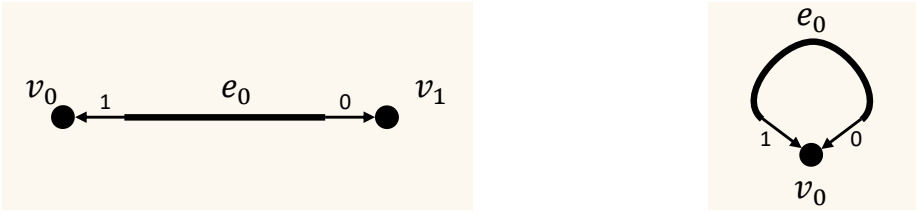
— une identification ou une désidentification n'induit pas toujours une relation d'inclusion entre les étapes t et $t+1$, dans un sens ou dans l'autre ;

Du fait de l'impossibilité de traiter les opérations d'identification et de désidentification, nous avons choisi de ne pas utiliser cette catégorie.

Exemple. Sur la figure 1.42 :

— l'ensemble semi-simplicial (K, d) n'est pas inclus dans (K', d') car v_1 n'appartient pas à K' ;

— l'ensemble semi-simplicial (K', d') n'est pas inclus dans (K, d) car $e_0 d'_0 = v_0$ et $e_0 d_0 = v_1$.



(a) (K, d) (b) (K', d')

FIGURE 1.42 – Identification de v_0 et v_1 sur ensemble semi-simplicial. (a) Avant identification. (b) Après identification.

Remarque. D'après [2], la persistance de tours est utilisable dans notre cadre d'étude sous certaines conditions. À notre connaissance, elle n'est définie que pour les complexes simpliciaux. Le Chapitre 4 de cette thèse fait le lien entre l'homologie effective et la persistance de tours. Au moment où [2] est paru, [16] travaillaient sur l'application de l'homologie effective à l'identification et la désidentification, dont le cadre est plus général à ce jour. Pour cette raison, nous avons choisi de continuer à explorer la piste de l'homologie effective, notamment dans le but d'étudier ses avantages, inconvénients et limites, dont les Chapitres 2 et 3 font l'objet.

1.4.4-c Homologie effective

Rappelons que l'Homologie Effective repose sur la notion de suite exacte courte effective (SECE) et du théorème qui leur est associé. Nous constatons que :

— l'évolution de l'objet subdivisé est prise en compte. Une SECE représente le passage de l'étape t à l'étape $t+1$;

— l'identification et la désidentification entrent dans le cadre de l'Homologie Effective.

Nous avons choisi d'explorer cette piste et d'étudier son application dans notre cadre d'étude.

Remarque. Mayer-Vietoris n'est pas utilisable avec toutes les identifications ou désidentifications (cf figure 1.42).

1.5 Théorème SECE pour l'identification et la désidentification

Cette section présente l'application du théorème SECE à l'identification et à la désidentification. Elle s'inspire de [16].

Remarque. Pour simplifier, nous choisissons les ensembles semi-simpliciaux comme type de structure pour cette section. Cette présentation se généralise à toute structure topologique sur laquelle sont définies l'identification et la désidentification, et à partir de laquelle on sait construire un complexe de chaînes.

Pour simplifier, nous utilisons le même symbole σ pour désigner un simplexe et le générateur qu'il engendre dans un complexe de chaînes.

1.5.1 Pour l'identification

Soient :

— un ensemble semi-simplicial (K^t, d^t) et le complexe de chaînes (C^t, ∂^t) qui lui est associé ;

— une identification caractérisée par I et $\zeta : K^t \to K^t$ de laquelle résulte (K^{t+1}, d^{t+1}) ;

— le complexe de chaînes $(C^{t+1}, \partial^{t+1})$ associé à (K^{t+1}, d^{t+1}) ;

— l'équivalence homologique

$$\gamma^t : (C^t, \partial^t) \overset{\rho^t}{\Longleftarrow} (C^{B,t}, \partial^{B,t}) \overset{\rho^{S,t}}{\Longrightarrow} (C^{S,t}, \partial^{S,t})$$

La suite exacte courte effective

$$(C, \partial) \underset{\mathsf{r}}{\overset{\mathsf{i}}{\rightleftarrows}} (C^t, \partial^t) \underset{\mathsf{s}}{\overset{\mathsf{j}}{\rightleftarrows}} (C^{t+1}, \partial^{t+1})$$

est définie par :

— C est engendré par les simplexes non-survivants de K^t ;

— $\sigma i = \sigma\zeta - \sigma$ pour tout $\sigma \in C$;

— $\sigma r = -\sigma$ si $\sigma\zeta \neq \sigma$, et $\sigma r = 0$ sinon, pour tout $\sigma \in C^t$;

— $\partial = i\partial^t r$ (cf [16], page 23) ;

— $\sigma j = \sigma\zeta$ pour tout $\sigma \in C^t$;

— $\sigma s = \sigma$ pour tout $\sigma \in C^{t+1}$;

L'équivalence homologique

$$\gamma : (C, \partial) \overset{\rho}{\Longleftarrow} (C^B, \partial^B) \overset{\rho^S}{\Longrightarrow} (C^S, \partial^S)$$

est calculée[13] à partir de (C, ∂). L'équivalence homologique γ^{t+1} est calculée par application du cas 1 du théorème SECE.

Remarque. L'opérateur de bord ∂ vérifie $\partial = i\partial^t r$ et peut être calculé comme tel, mais cela suppose que i et r soient connus. Dans [16], page 23, un algorithme est donné pour construire ∂ à partir de d^t, I et ζ.

13. Une façon de faire est de choisir une réduction identité pour ρ, et de composer des réductions élémentaires sur $(C^B, \partial^B) = (C, \partial)$ pour calculer ρ^S (cf. annexes A et B). Une autre façon de faire pour calculer ρ^S consiste à utiliser la notion d'HDVF présentée dans [80].

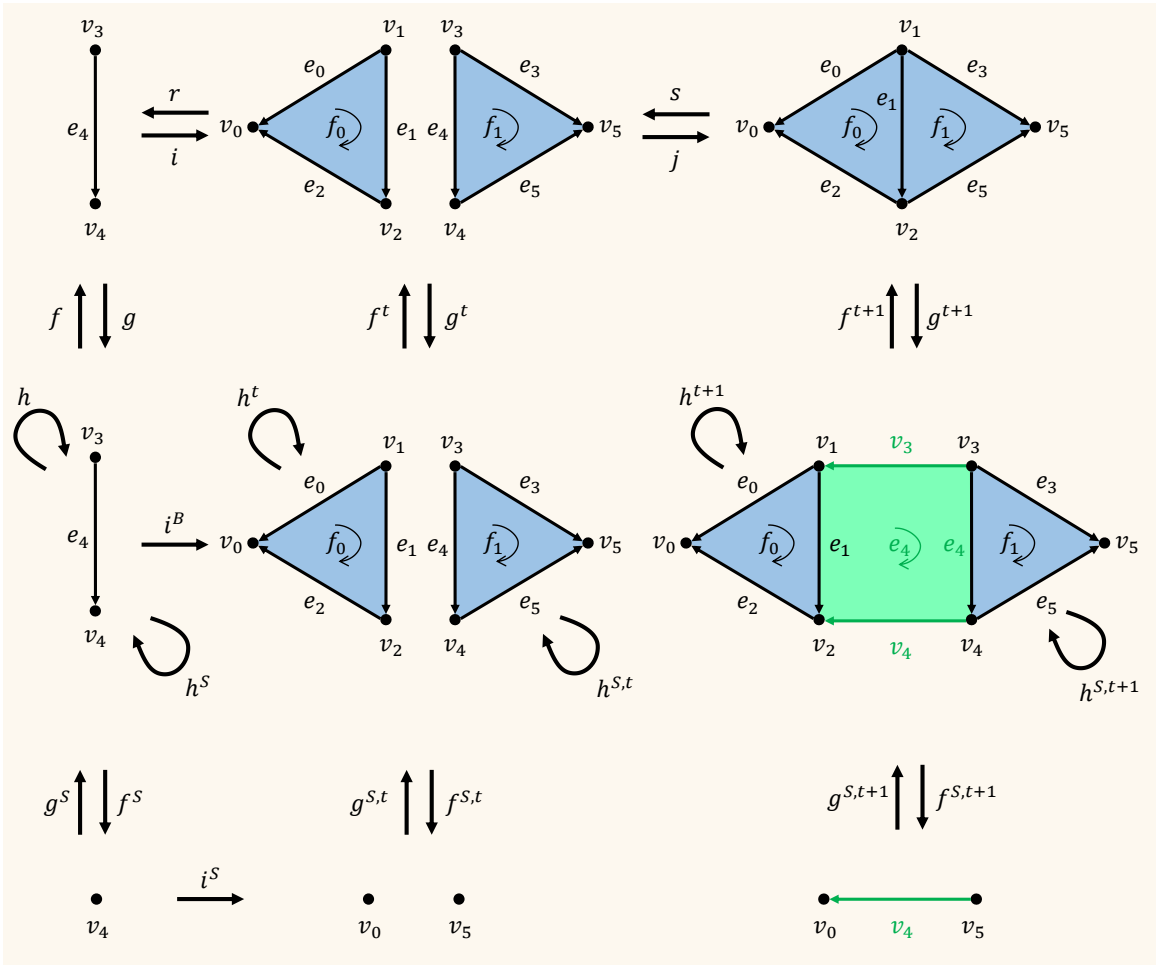


FIGURE 1.43 – Théorème SECE appliqué à une identification. À gauche, l'équivalence homologique γ. Au milieu, l'équivalence homologique γ^t. À droite, l'équivalence homologique γ^{t+1}. En vert, les cellules de $C^{B,t+1}$ issues de C^B et les cellules de $C^{S,t+1}$ issues de C^S.

Exemple. La figure 1.43 illustre une application du théorème SECE pour une identification caractérisée par :

$$I^0 = \{\{v_0\}, \{v_1, v_3\}, \{v_2, v_4\}, \{v_5\}\}$$
$$I^1 = \{\{e_0\}, \{e_1, e_4\}, \{e_2\}, \{e_3\}, \{e_5\}\}$$
$$I^2 = \{\{f_0\}, \{f_1\}\}$$

et $\zeta : K^t \to K^t$:

— $v_0\zeta = v_0$, $v_1\zeta = v_1$, $v_3\zeta = v_1$, $v_2\zeta = v_2$, $v_4\zeta = v_2$ et $v_5\zeta = v_5$;

— $e_0\zeta = e_0$, $e_1\zeta = e_1$, $e_4\zeta = e_1$, $e_2\zeta = e_2$, $e_3\zeta = e_3$, $e_5\zeta = e_5$;

— $f_0\zeta = f_0$, $f_1\zeta = f_1$.

Les morphismes i, j, r et s sont ceux explicités dans le tableau de l'exemple de la définition 1.4.6. En particulier :

— $v_3 i = v_1 - v_3$;

— $(v_1 - v_3)j = v_1 - v_1 = 0$;
— $(v_1 - v_3)r = v_3$;
— $v_1 s = v_1$.

Les morphismes i^B et i^S sont définis par :

$i^B : C^B \to C^{B,t}$	$i^S : C^S \to C^{S,t}$
$v_3 i^B = v_1 - v_3$ $v_4 i^B = v_2 - v_4$ $e_4 i^B = e_1 - e_4$	$v_4 i^S = v_0 - v_5$

1.5.2 Pour la désidentification

Soient :

— un ensemble semi-simplicial (K^t, d^t) et le complexe de chaînes (C^t, ∂^t) qui lui est associé ;
— une désidentification dont l'identification inverse est caractérisée par I et $\zeta : K^{t+1} \to K^{t+1}$, de laquelle résulte (K^t, ∂^t) ;
— le complexe de chaînes $(C^{t+1}, \partial^{t+1})$ associé à $(K^{t+1}, \partial^{t+1})$;
— l'équivalence homologique

$$\gamma^t : (C^t, \partial^t) \overset{\rho^t}{\Longleftarrow} (C^{B,t}, \partial^{B,t}) \overset{\rho^{S,t}}{\Longrightarrow} (C^{S,t}, \partial^{S,t})$$

La suite exacte courte effective

$$(C, \partial) \underset{\mathsf{r}}{\overset{\mathsf{i}}{\rightleftarrows}} (C^{t+1}, \partial^{t+1}) \underset{\mathsf{s}}{\overset{\mathsf{j}}{\rightleftarrows}} (C^t, \partial^t)$$

est définie de la même façon que dans le cas de l'identification, en échangeant les exposants t et $t+1$. L'équivalence homologique γ est calculée à partir de (C, ∂), là aussi de la même façon que dans le cas de l'identification. L'équivalence homologique γ^{t+1} est calculée par application du cas 2 du théorème SECE.

Exemple. La figure 1.44 illustre une application du théorème SECE pour une désidentification. L'identification inverse est caractérisée par I et $\zeta : K^{t+1} \to K^{t+1}$ identiques à ceux présentés dans l'exemple de la sous-section précédente 1.5.1. En conséquence, les morphismes i,j,r et s sont aussi ceux présentés dans l'exemple de la définition 1.4.6. Les morphismes χ, χ^B et χ^S sont définis par :

$\chi : C^t \to C$	$\chi^B : C^{B,t} \to C^B$	$\chi^S : C^{S,t} \to C^S$
$v_0\chi = 0$	$v_0\chi^B = 0$	$v_0\chi^S = 0$
$v_1\chi = 0$	$v_1\chi^B = 0$	
$v_2\chi = 0$	$v_2\chi^B = 0$	
$v_5\chi = 0$	$v_5\chi^B = 0$	
$e_0\chi = 0$	$e_0\chi^B = 0$	
$e_1\chi = 0$	$e_1\chi^B = 0$	
$e_2\chi = 0$	$e_2\chi^B = 0$	
$e_3\chi = v_3$	$e_3\chi^B = v_3$	
$e_5\chi = v_4$	$e_5\chi^B = v_5$	
$f_0\chi = 0$	$f_0\chi^B = 0$	
$f_1\chi = e_4$	$f_1\chi^B = e_4$	

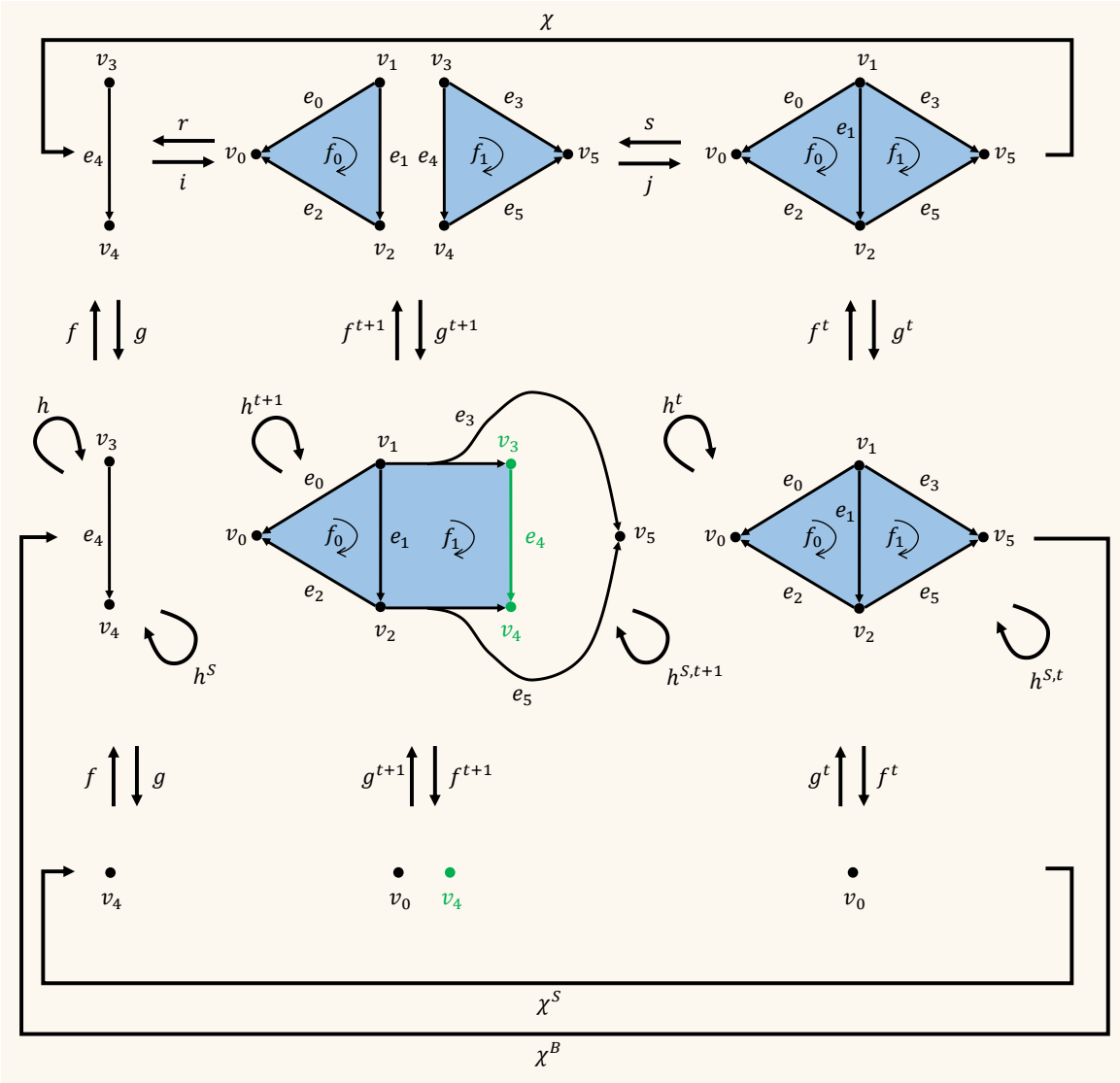


FIGURE 1.44 – Théorème SECE appliqué à une désidentification. À gauche, l'équivalence homologique γ. Au milieu, l'équivalence homologique γ^{t+1}. À droite, l'équivalence homologique γ^t. En vert, les cellules de $C^{B,t+1}$ issues de C^B et les cellules de $C^{S,t+1}$ issues de C^S.

Chapitre 2

Calcul incrémental séquentiel

Contenu du chapitre

2.1 Analyse

Nous avons vu en section 1.5 que le théorème des suites exactes courtes effectives (théorème SECE) s'applique aux opérations d'identification et de désidentification. En particulier, il peut être utilisé pour calculer les variations d'homologie d'une structure topologique qui évolue dans un processus de construction où le passage d'une étape de construction t à une étape de construction $t+1$ se fait par identification ou désidentification. Le théorème SECE est utilisé pour maintenir une équivalence homologique au fil des étapes de construction. Dans cette section, nous analysons le calcul de l'équivalence homologique γ^{t+1} (de l'étape de construction $t+1$), étant données :

— l'équivalence homologique γ^t (de l'étape de construction t) ;

— la SECE représentant l'opération de passage de t à $t+1$.

Pour cela, nous représentons les morphismes sous forme de matrices à coefficients dans $\mathbb{Z}$, et nous étudions les caractéristiques et le calcul de ces dernières. L'objectif est de mettre en évidence les particularités de ces calculs dans le but de minimiser leur temps. La section se compose de plusieurs sous-sections :

1. le préambule, dans lequel nous posons le cadre de l'analyse ;
2. l'analyse d'une SECE induite par une identification ou une désidentification ;
3. l'analyse du calcul de γ^{t+1} lorsque le passage de t à $t+1$ se fait par identification ;
4. l'analyse du calcul de γ^{t+1} lorsque le passage de t à $t+1$ se fait par désidentification.

2.1.1 Préambule

Dans ce préambule, nous présentons le contexte et les objectifs de l'analyse, puis nous explicitons notre choix de représentation matricielle d'un morphisme.

2.1.1-a Contexte

Pour rappel, les travaux présentés dans cette thèse consistent à étudier la mise en œuvre du théorème SECE qui permet, à priori, de **calculer incrémentalement les variations d'homologie** d'une structure topologique qui évolue dans un **processus de construction**. Plus précisément, le calcul consiste à maintenir, au fil du processus de construction, une équivalence homologique γ^t associée à la structure topologique à l'étape de construction t. γ^t contient notamment un "petit" complexe de chaînes sur lequel l'homologie de la structure topologique est calculée. Le théorème SECE est utilisé pour maintenir γ^t au fil des **étapes de construction**, en supposant que le passage de l'étape t à l'étape $t+1$ se fait soit par identification, soit par désidentification. L'opération est symbolisée par une SECE (cf. section 1.5). Nous analysons l'application du théorème SECE à ces opérations afin de mettre en évidence les **prérequis d'implémentation** qui doivent être satisfaits pour exploiter le fait que γ^{t+1} **est une modification de** γ^t. Dit autrement, on souhaite optimiser le calcul de γ^{t+1} en réutilisant au maximum les données de γ^t.

En toute généralité, étudier l'application du théorème SECE à l'identification ou la désidentification ne fait pas sens du fait de la diversité des cas à considérer. Par exemple, on peut supposer que le processus de construction n'est composé que d'une seule identification dans laquelle la totalité des cellules de la structure topologique sont identifiées. Dans ce cas, utiliser le théorème SECE pour calculer incrémentalement les variations d'homologie de la structure topologique n'est pas pertinent. Nous supposons donc que le processus de construction est composé

de plusieurs opérations, et que ces dernières ne modifient qu'une "petite" partie la structure topologique. Nous émettons les hypothèses ci-dessous pour la totalité du chapitre.

Hypothèses.

(i) L'équivalence homologique γ^{t+1} est calculée par modification de l'équivalence homologique γ^t.

(ii) Le nombre de cellules impactées par l'opération de passage de l'étape t à l'étape $t+1$ du processus de construction est "petit" comparé au nombre total de cellules de la structure topologique.

2.1.1-b Objectifs et résultats

L'analyse consiste à étudier les caractéristiques et les calculs des matrices utilisés dans l'application du théorème SECE à l'identification et à la désidentification. Les objectifs sont de mettre en évidence :

— les particularités des calculs. Sont-ils plus efficaces s'ils sont effectués d'une certaine façon ? Pourquoi ?

— les prérequis que doit satisfaire une implémentation pour optimiser le calcul de γ^{t+1}. Quelle structure de données utiliser ? Quelles opérations implémenter ?

En tenant compte des hypothèses établies précédemment, on constate à l'issue de l'analyse que deux cas se distinguent :

— les cas "limités", où seule une partie des matrices de γ^t est maintenue ;

— le cas général, où toutes les matrices de γ^t sont maintenues.

Pour les cas "limités", on observe que :

— la complexité en temps du calcul de γ^{t+1} dépend du nombre de cellules impactées par l'opération de passage de t à $t+1$, et de leur étoile "directe" [1] ;

— la complexité en espace de γ^{t+1} est égale à la complexité en espace de $\gamma^{t=0}$ à laquelle s'ajoute la somme des variations en espace des étapes précédentes. À chaque étape, cette variation dépend en partie du bord des cellules ajoutées dans les complexes de chaînes de γ^{t+1}, et en partie de leur comportement dans les réductions ρ^{t+1} et $\rho^{S,t+1}$;

Pour le cas général, on observe le même type de résultat sauf pour le calcul de $h^{S,t+1}$, qui, dans le pire des cas, est lié au nombre de cellules de la structure topologique.

2.1.1-c Représentation matricielle d'un morphisme

Nous supposons que tous les morphismes manipulés sont représentés sous la forme matricielle décrite dans cette sous-section. Soit un morphisme $\phi : X \to Y$ où X et Y sont deux groupes abéliens libres [2] de bases respectives $B_X = \{x_0, x_1, \ldots, x_q\}$ et $B_Y = \{y_0, y_1, \ldots, y_{q'}\}$, avec $q, q' \in \mathbb{N}$. En particulier, pour tout x de X :

$$x = \sum_{i=0}^{q} \lambda_i x_i \qquad \forall i | 0 \leq i \leq q, x_i \in B_X, \lambda_i \in \mathbb{Z}$$

1. L'étoile directe d'une p-cellule est son étoile restreinte aux $(p+1)$-cellules.
2. Tous les groupes abéliens considérés dans cette thèse sont libres, c'est-à-dire qu'ils possèdent une base.

$$x_i\phi = \sum_{j=0}^{q'} \lambda_{i,j} y_j \qquad \forall j | 0 \leq j \leq q', y_j \in B_Y, \lambda_{i,j} \in \mathbb{Z}$$

Convention. La représentation matricielle de ϕ, est telle que :

— chaque ligne correspond à un élément de B_X ;

— chaque colonne correspond à un élément de B_Y ;

— la case $\phi[i,j]$ est égale à $\lambda_{i,j}$.

La matrice ϕ est représentée par :

$$\begin{array}{c} \phi \quad B_Y \\ B_X \begin{bmatrix} \mathbb{Z} \end{bmatrix} \end{array}$$

Remarque. Cette représentation est souvent utilisée dans la littérature [81, 53, 82, 83], parfois avec des coefficients dans $\mathbb{Z}_2 = \mathbb{Z}/2\mathbb{Z}$. Elle implique que les dimensions de la matrice ϕ peuvent être égales à 0 si B_X et/ou B_Y est vide, ce qui est parfois le cas [3]. Par la suite :

— $\phi[x_i, y_j]$ désigne $\phi[i,j]$, c'est-à-dire $\lambda_{i,j}$;

— nous utilisons parfois la notation $\phi : X \to Y$ pour désigner la matrice ϕ dont les lignes représentent B_X et les colonnes représentent B_Y.

Valeurs explicites. Les matrices manipulées dans l'application du théorème SECE à l'identification et la désidentification sont toujours des variations soit de la matrice nulle, soit de l'identité. Par exemple :

— une matrice de bord est généralement proche de la matrice nulle. Intuitivement, cela découle du fait qu'une ligne de cette matrice représente le bord d'une cellule, et que le nombre de cellules dans le bord d'une cellule est généralement petit par rapport au nombre total de cellules de la structure topologique. Dit autrement, il y a autant de coefficients 0 dans une ligne d'une matrice de bord que de cellules qui ne sont pas dans le bord de la cellule représentée par cette ligne ;

— le morphisme j a pour domaine les cellules d'une structure topologique avant une identification, et pour codomaine les cellules de la même structure topologique après cette identification. En particulier, j est l'identité pour toutes les cellules sauf pour celles qui ne survivent pas à l'identification. Or, d'après l'hypothèse (ii), le nombre de cellules non survivantes est "petit" par rapport au nombre total de cellules de la structure topologique, donc j est très proche de l'identité.

Pour une matrice ϕ donnée, nous étudions sa variation V par rapport à l'identité ou par rapport à la matrice nulle. Dit autrement, ϕ se décompose en :

— $\phi = [0] + V$ lorsque l'on étudie sa variation par rapport à la matrice nulle (donc $V = \phi$) ;

— $\phi = [1] + V$ lorsque l'on étudie sa variation par rapport à "l'identité", c'est-à-dire où les cases $\phi[x,x] = 1$ sont implicites, avec $x \in B_X \cap B_Y$.

3. La matrice représentant $\partial^0 : C^0 \to \{0\}$ a 0 colonnes.

Nous notons ϕ_Δ le nombre de valeurs explicites de ϕ, c'est-à-dire le nombre de valeurs non nulles de V. Par la suite, une valeur nulle d'une matrice est représentée par un point.

Remarque. Il ne faut pas confondre variation par rapport à l'identité et variation par rapport à la matrice identité, qui implique notamment $B_X = B_Y$.

Exemple 1. Soient $B_X = B_Y = \{a, b, c\}$, et un morphisme $\phi : X \to Y$ défini par :

— $a\phi = 3a + b$;

— $b\phi = b$;

— $c\phi = 2a - c$;

La matrice ϕ est :

$$\begin{array}{c} \phi \\ a \\ b \\ c \end{array} \begin{array}{c} \begin{array}{ccc} a & b & c \end{array} \\ \begin{bmatrix} 3 & 1 & . \\ . & 1 & . \\ 2 & . & -1 \end{bmatrix} \end{array}$$

En particulier, étudier la variation de ϕ par rapport à l'identité revient à étudier V dans :

$$\phi = \begin{array}{c} [1] \\ a \\ b \\ c \end{array} \begin{array}{c} \begin{array}{ccc} a & b & c \end{array} \\ \begin{bmatrix} . & . & . \\ . & 1 & . \\ . & . & . \end{bmatrix} \end{array} + \begin{array}{c} V \\ a \\ b \\ c \end{array} \begin{array}{c} \begin{array}{ccc} a & b & c \end{array} \\ \begin{bmatrix} 3 & 1 & . \\ . & . & . \\ 2 & . & -1 \end{bmatrix} \end{array}$$

Dans ce cas, $\phi_\Delta = 4$. Étudier la variation de ϕ par rapport à la matrice nulle revient à étudier V dans :

$$\phi = \begin{array}{c} [0] \\ a \\ b \\ c \end{array} \begin{array}{c} \begin{array}{ccc} a & b & c \end{array} \\ \begin{bmatrix} . & . & . \\ . & . & . \\ . & . & . \end{bmatrix} \end{array} + \begin{array}{c} V \\ a \\ b \\ c \end{array} \begin{array}{c} \begin{array}{ccc} a & b & c \end{array} \\ \begin{bmatrix} 3 & 1 & . \\ . & 1 & . \\ 2 & . & -1 \end{bmatrix} \end{array}$$

Dans ce cas, $\phi_\Delta = 5$.

Exemple 2. Soient $B_X = \{a, b, c\}, B_Y = \{e, b, f, a\}$ et un morphisme $\phi : X \to Y$ défini par :

— $a\phi = 3e + a$;

— $b\phi = e + b + 2f$;

— $c\phi = 2e + 3a$;

La matrice ϕ est :

$$\begin{array}{c} \phi \\ a \\ b \\ c \end{array} \begin{array}{c} \begin{array}{cccc} e & b & f & a \end{array} \\ \begin{bmatrix} 3 & . & . & 1 \\ 1 & 1 & 2 & . \\ 2 & . & . & 3 \end{bmatrix} \end{array}$$

En particulier, étudier la variation de ϕ par rapport à l'identité revient à étudier V dans :

$$\phi = \begin{array}{c} \\ a \\ b \\ c \end{array} \begin{array}{c} \begin{array}{cccc} [1] & e & b & f & a \end{array} \\ \begin{bmatrix} . & . & . & 1 \\ . & 1 & . & . \\ . & . & . & . \end{bmatrix} \end{array} + \begin{array}{c} \\ a \\ b \\ c \end{array} \begin{array}{c} \begin{array}{cccc} V & e & b & f & a \end{array} \\ \begin{bmatrix} 3 & . & . & . \\ 1 & . & 2 & . \\ 2 & . & . & 3 \end{bmatrix} \end{array}$$

Dans ce cas, $\phi_\Delta = 5$. Étudier la variation de ϕ par rapport à la matrice nulle revient à étudier V dans :

$$\phi = \begin{array}{c} \\ a \\ b \\ c \end{array} \begin{array}{c} \begin{array}{cccc} [0] & e & b & f & a \end{array} \\ \begin{bmatrix} . & . & . & . \\ . & . & . & . \\ . & . & . & . \end{bmatrix} \end{array} + \begin{array}{c} \\ a \\ b \\ c \end{array} \begin{array}{c} \begin{array}{cccc} V & e & b & f & a \end{array} \\ \begin{bmatrix} 3 & . & . & 1 \\ 1 & 1 & 2 & . \\ 2 & . & . & 3 \end{bmatrix} \end{array}$$

Dans ce cas, $\phi_\Delta = 7$.

Caractéristiques matricielles étudiées. Par la suite, pour chaque matrice considérée, les caractéristiques étudiées sont :

— ses nombres de lignes et colonnes ;

— le fait qu'elle soit proche de la matrice nulle ou de l'identité ;

— son nombre de valeurs explicites.

2.1.1-d À propos du produit matriciel

Le **produit matriciel** est un calcul souvent utilisé dans l'application du théorème SECE. En toute généralité, l'ordre de grandeur de la complexité en temps du produit matriciel est [4] n^3, où n désigne les dimensions des matrices dont on fait le produit (en supposant qu'elles soient carrées). Ce niveau d'abstraction est problématique pour l'analyse : il ne rend pas compte des caractéristiques des matrices manipulées. Du fait de l'hypothèse (ii), les produits matriciels que nous analysons impliquent la plupart du temps une matrice dont l'une des dimensions est "petite" par rapport aux autres dimensions. Elle est appelée **dimension exploitable**. Tout l'enjeu pour l'optimisation du calcul de γ^{t+1} est alors d'utiliser cette dimension pour réduire au maximum le temps de calcul du produit. De plus, nous supposons que **toutes les matrices que nous manipulons sont creuses**. En particulier, le temps de calcul du produit de deux matrices creuses dépend entièrement de la dimension exploitable, et de leurs nombres maximum de valeurs explicites par lignes ou par colonnes.

Remarque. Afin de conserver une certaine simplicité dans les exemples qui suivent, nous comparons les produits seulement selon les nombres d'additions et de multiplications qu'ils requièrent. Cette comparaison fait abstraction du temps d'accès au premier élément d'une ligne ou d'une colonne, que l'on suppose constant. Nous rappelons que le produit matriciel naïf, avec des matrices denses, nécessite n^3 multiplications et $n^2(n-1)$ additions.

4. Il existe des algorithmes de produit matriciel dont les ordres de grandeur de la complexité en temps sont inférieurs à n^3, comme [84]. En pratique, ces algorithmes sont inutilisables du fait des constantes qui accompagnent les ordres de grandeur annoncés.

A	e	f	g	h
a	.	1	5	.
b	.	4	3	.
c	.	.	.	.
d	.	2	7	.

$*$

B	i	j
e	1	2
f	-5	6
g	1	1
h	.	.

$=$

AB	i	j
a	.	11
b	-17	27
c	.	.
d	-3	19

(a) Produit matriciel naïf.

A	e	f	g	h
a	.	1	5	.
b	.	4	3	.
c	.	.	.	.
d	.	2	7	.

$*$

B	i	j
e	1	2
f	-5	6
g	1	1
h	.	.

$=$

AB	i	j
a	.	11
b	-17	27
c	.	.
d	-3	19

(b) Produit matriciel creux en parcourant les lignes de A.

A	e	f	g	h
a	.	1	5	.
b	.	4	3	.
c	.	.	.	.
d	.	2	7	.

$*$

B	i	j
e	1	2
f	-5	6
g	1	1
h	.	.

$=$

AB	i	j
a	.	11
b	-17	27
c	.	.
d	-3	19

(c) Produit matriciel creux en parcourant les colonnes de B.

FIGURE 2.1 – Le produit de deux matrices A et B réalisé de différentes manières : en vert, les lignes, colonnes, et cases manipulées dans le produit.

Exemple 1. Dans le cas de la figure 2.1a, le produit est naïf. Les matrices sont traitées comme des matrices denses, et AB est calculée en $4 * 2 * 4 = 32$ multiplications et $4 * 2 * 3 = 24$ additions.

Exemple 2. Dans le cas de la figure 2.1b, le produit tient compte du fait que les matrices sont creuses mais il ne tient pas compte du fait que B a 2 colonnes. Dans ce cas, AB est calculée en 12 multiplications et 6 additions. Les 4 lignes de A sont parcourues, et :

— pour la ligne a, les opérations sont, dans l'ordre,
  · $AB[a,i] = A[a,f] * B[f,i] = 1 * (-5) = -5$;
  · $AB[a,j] = A[a,f] * B[f,j] = 1 * 6 = 6$;
  · $AB[a,i] = AB[a,i] + A[a,g] * B[g,i] = (-5) + 5 * 1 = 0$;
  · $AB[a,j] = AB[a,j] + A[a,g] * B[g,j] = 6 + 5 * 1 = 11$;
— pour la ligne b, les opérations sont, dans l'ordre,
  · $AB[b,i] = A[b,f] * B[f,i] = 4 * (-5) = -20$;
  · $AB[b,j] = A[b,f] * B[f,j] = 4 * 6 = 24$;
  · $AB[b,i] = AB[b,i] + A[b,g] * B[g,i] = (-20) + 3 * 1 = -17$;
  · $AB[b,j] = AB[b,j] + A[b,g] * B[g,j] = 24 + 3 * 1 = 27$;
— pour la ligne c, aucune opération arithmétique ;
— pour la ligne d, les opérations sont, dans l'ordre,
  · $AB[d,i] = A[d,f] * B[f,i] = 2 * (-5) = -10$;
  · $AB[d,j] = A[d,f] * B[f,j] = 2 * 6 = 12$;
  · $AB[d,i] = AB[d,i] + A[d,g] * B[g,i] = (-10) + 7 * 1 = -3$;
  · $AB[d,j] = AB[d,j] + A[d,g] * B[g,j] = 12 + 7 * 1 = 19$;

En particulier, observons que le produit se fait en parcourant les 4 lignes de A, et que, pour chacune d'elles, 2 lignes de B sont utilisées car A a 2 valeurs non nulles par ligne.

Exemple 3. Dans le cas de la figure 2.1c, le produit utilise le fait que les matrices sont creuses et tient compte du fait que B a 2 colonnes. AB est calculée en 12 multiplications et 6 additions. Les 2 colonnes de B sont parcourues, et :

— pour la colonne i, les opérations sont, dans l'ordre,
  · $AB[a,i] = B[f,i] * A[a,f] = (-5) * 1 = -5$;
  · $AB[b,i] = B[f,i] * A[b,f] = (-5) * 4 = -20$;
  · $AB[d,i] = B[f,i] * A[d,f] = (-5) * 2 = -10$;
  · $AB[a,i] = AB[a,i] + B[g,i] * A[a,g] = (-5) + 1 * 5 = 0$;
  · $AB[b,i] = AB[b,i] + B[g,i] * A[b,g] = (-20) + 1 * 3 = -17$;
  · $AB[d,i] = AB[d,i] + B[g,i] * A[d,g] = (-10) + 1 * 7 = -3$;
— pour la colonne j, les opérations sont, dans l'ordre,
  · $AB[a,j] = B[f,j] * A[a,f] = 6 * 1 = 6$;
  · $AB[b,j] = B[f,j] * A[b,f] = 6 * 4 = 24$;
  · $AB[d,j] = B[f,j] * A[d,f] = 6 * 2 = 12$;
  · $AB[a,j] = AB[a,j] + B[g,j] * A[a,g] = 6 + 1 * 5 = 11$;
  · $AB[b,j] = AB[b,j] + B[g,j] * A[b,g] = 24 + 1 * 3 = 27$;
  · $AB[d,j] = AB[d,j] + B[g,j] * A[d,g] = 12 + 1 * 7 = 19$;

Par rapport à l'exemple 2, on note donc le même nombre d'opérations arithmétiques. En particulier, observons que le produit se fait en parcourant les 2 colonnes de B, et que, pour chacune d'elles, 3 colonnes de A sont utilisées car B a 3 valeurs non nulles par colonne.

Un autre calcul souvent utilisé est le **produit matriciel en chaîne**, c'est-à-dire où une matrice est calculée comme étant le produit de plus de deux matrices. Dans ce cas, les dimensions des matrices impliquées et **l'ordre dans lequel se fait le produit** jouent un rôle déterminant sur l'efficacité du calcul. Dit autrement, **le temps de calcul de $A(BC)$ peut être différent du temps de calcul de $(AB)C$.**

Exemple. Soit une matrice A de dimension 2×3, une matrice B de dimension 3×80 et une matrice C de dimension 80×1. Considérons le produit ABC :

$$\overset{A\ \ 3}{2\left[\mathbb{Z}\right]} \quad * \quad \overset{B\ \ 80}{3\left[\mathbb{Z}\right]} \quad * \quad \overset{C\ \ 1}{80\left[\mathbb{Z}\right]}$$

Il y a deux façons de faire :

— $(AB)C$. Dans ce cas, le produit matriciel naïf nécessite par exemple $2 * 3 * 80 = 480$ multiplications pour calculer (AB) et $2 * 80 * 1 = 160$ multiplications pour calculer $(AB)C$, soit 640 multiplications en tout ;

— $A(BC)$. Dans ce cas, le produit matriciel naïf nécessite par exemple $3 * 80 * 1 = 240$ multiplications pour calculer (BC) et $2 * 3 * 1 = 6$ multiplications pour calculer $A(BC)$, soit 246 multiplications en tout.

2.1.1-e À propos de la fusion de matrice

La **fusion de matrices** est un calcul souvent utilisé dans l'application du théorème SECE. Soit quatre matrices A, B, C et D. La fusion de matrices consiste à **modifier** une de ces matrices, nommée X, en lui ajoutant un nombre de lignes, colonnes, et valeurs explicites qui va dépendre des 3 autres matrices fusionnées. Ce calcul est représenté par :

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

Pour rappel, nous supposons que **toutes les matrices que nous manipulons sont creuses**. En conséquence, le temps de calcul de la fusion de matrices dépend entièrement des dimensions des 3 matrices fusionnées, et de leur nombre de valeurs explicites. De plus, du fait de l'hypothèse (ii), les matrices fusionnées ont la plupart du temps au moins une "petite" dimension. Tout l'enjeu pour l'optimisation du calcul de γ^{t+1} est alors **d'exploiter ces dimensions** pour réduire au maximum le temps de calcul de la fusion de matrice.

Remarque. La fusion de 2 matrices est un cas particulier de la fusion de 4 matrices, où deux des matrices sont des matrices avec une dimension nulle, et sans valeurs explicites. Par exemple, $A = \begin{bmatrix} A & B \end{bmatrix}$ revient à considérer que C et D ont 0 lignes, et autant de colonnes qu'en comptent A et B.

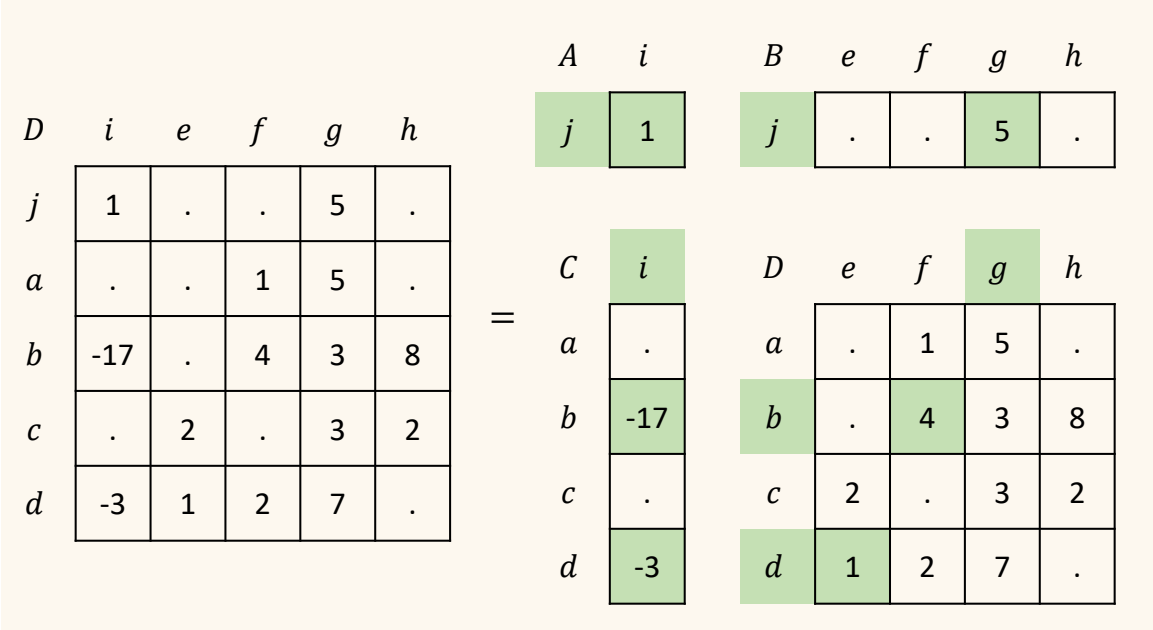


FIGURE 2.2 – Fusion de matrices creuses par modification de D (donc $X = D$). En vert, les lignes, colonnes, et cases auxquelles on accède.

Exemple. La figure 2.2 illustre une fusion de matrices $D = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$. Le nombre de lignes de B représente sa dimension exploitable. Le nombre de colonnes de C représente sa dimension exploitable. En conséquence, les deux dimensions de A sont exploitables. Elle est calculée :

- en ajoutant la ligne j de B à D ;
- en ajoutant la ligne j de A à C ;
- en ajoutant la colonne i de C à D.

2.1.2 Suite exacte courte effective

Nous avons vu en section 1.5 qu'étant donnée une identification ou une désidentification sur une structure topologique, on peut calculer une SECE :

$$(C, \partial) \underset{\text{r}}{\overset{\text{i}}{\rightleftarrows}} (C', \partial') \underset{\text{s}}{\overset{\text{j}}{\rightleftarrows}} (C'', \partial'')$$

Dans cette thèse, un complexe de chaînes est toujours associé à une structure topologique. Pour simplifier, la terminologie et les notations de l'identification sont étendues aux complexes de chaînes. De plus, le calcul des complexes de chaînes (C', ∂') et (C'', ∂'') dépend de la nature de l'opération qui induit la SECE :

- dans le cas de l'identification, (C'', ∂'') est une modification de (C', ∂'). Dit autrement, (C', ∂') correspond au complexe de chaînes (C^t, ∂^t) de l'étape de construction t et (C'', ∂'') correspond au complexe de chaînes $(C^{t+1}, \partial^{t+1})$ de l'étape de construction $t+1$;
- dans le cas de la désidentification, (C', ∂') est une modification de (C'', ∂''). Dit autrement, (C'', ∂'') correspond au complexe de chaînes (C^t, ∂^t) de l'étape de construction t et (C', ∂') correspond au complexe de chaînes $(C^{t+1}, \partial^{t+1})$ de l'étape de construction $t+1$.

Rappelons que la désidentification induit une identification inverse, à partir de laquelle la SECE est calculée. Dit autrement, pour analyser le calcul de cette SECE dans le cas de l'identification et de la désidentification, il suffit d'analyser son calcul pour une identification caractérisée par une famille I de partitions d'une base $B_{C'}$ de C' et par $\zeta : B_{C'} \to B_{C''}$. $B_{C'}$ correspond aux cellules de la structure topologique avant identification et $B_{C''}$ correspond aux cellules de la structure topologique après identification, dites survivantes, d'où $B_{C''} \subseteq B_{C'}$. C est engendré par les cellules non-survivantes de $B_{C'}$ donc $|B_C| = |B_{C'}| - |B_{C''}|$, où B_C est une base de C. De plus, d'après l'hypothèse (ii), B_C est "petite" par rapport à $B_{C'}$ et $B_{C''}$. Dans cette sous-section, nous analysons le calcul des matrices i, j, r et s de la SECE induite par cette identification, ainsi que le calcul de l'équivalence homologique γ associée à (C, ∂).

Remarque. L'étude qui suit est systématique et l'on retrouve certains mécanismes similaires pour plusieurs matrices. Il est important que l'étude soit exhaustive pour servir de référence aux sections qui suivent et garantir que tous les prérequis d'implémentation sont mis en évidence.

2.1.2-a Analyse du calcul de i, r, j, s.

Matrice i. Rappelons que pour toute cellule σ de B_C, $\sigma i = \sigma\zeta - \sigma$, où $\sigma\zeta$ et σ appartiennent à $B_{C'}$. En conséquence, toute ligne de i est composée de strictement 2 valeurs explicites. Les caractéristiques matricielles de i sont :

- $|B_C|$ lignes et $|B_{C'}|$ colonnes. Or, $|B_C|$ est "petit" par rapport à $|B_{C'}|$, d'après l'hypothèse (ii) du paragraphe 2.1.1-a ;
- on l'étudie comme une variation de la matrice nulle ;
- $i_\Delta = 2|B_C|$. Le nombre de valeurs explicites de i est donc fonction de $|B_C|$, et est donc "petit".

La matrice i est construite pour chaque opération de passage de l'étape de construction t à $t+1$. Observons que, pour éviter de dépendre du nombre de cellules de la structure topologique à l'étape t, **sa construction ne doit pas dépendre de ses $|B_{C'}|$ colonnes**.

Remarque. Le nombre de valeurs explicites i_Δ suppose que i est représentée sous forme de matrice creuse, ce qui induit un prérequis d'implémentation. On retrouve ce prérequis pour toutes les matrices de l'analyse, qu'elles soient étudiées comme une variation de la matrice nulle ou de l'identité (cf. paragraphe 2.1.1-c). Pour simplifier, ce prérequis est mentionné sans être motivé par la suite.

Prérequis d'implémentation.

- Représentation en matrice creuse : les valeurs nulles sont implicites ;
- Construction d'une matrice qui ne dépend pas de son nombre de colonnes.

Matrice r. Rappelons que pour toute cellule σ de $B_{C'}$, $\sigma r = -\sigma$ si $\sigma \neq \sigma\zeta$, et $\sigma r = 0$ sinon. De plus, les cellules de $B_{C'}$ pour lesquelles $\sigma \neq \sigma\zeta$ sont les cellules non-survivantes. En conséquence il y a $|B_C| = |B_{C'}| - |B_{C''}|$ lignes de r composées de strictement 1 valeur explicite. Les caractéristiques matricielles de r sont :

- $|B_{C'}|$ lignes et $|B_C|$ colonnes. Or, $|B_C|$ est "petit" par rapport à $|B_{C'}|$, d'après l'hypothèse (ii) du paragraphe 2.1.1-a ;
- on l'étudie comme une variation de la matrice nulle ;

— $r_\Delta = |B_C|$. Le nombre de valeurs explicites de r est donc fonction de $|B_C|$, et est donc "petit".

La matrice r est construite pour chaque opération de passage de l'étape de construction t à $t+1$. Observons que pour éviter de dépendre du nombre de cellules de la structure topologique à l'étape t, **sa construction ne doit pas dépendre de ses $|B_{C'}|$ lignes**.

Prérequis d'implémentation.

— Représentation en matrice creuse : les valeurs nulles sont implicites ;

— Construction d'une matrice qui ne dépend pas de son nombre de lignes.

Matrice j. Rappelons que pour toute cellule σ de $B_{C'}$, $\sigma j = \sigma\zeta$ avec $\sigma\zeta$ appartenant à $B_{C''}$. Dit autrement, j **est l'identité pour toute cellule survivante de** $B_{C'}$. En conséquence, aucune ligne de j ne contient de valeurs explicites sauf les $|B_{C'}| - |B_{C''}|$ lignes représentant une cellule non-survivante de $B_{C'}$, qui sont composées d'exactement 1 valeur explicite. Les caractéristiques matricielles de j sont :

— $|B_{C'}|$ lignes et $|B_{C''}|$ colonnes ;

— on l'étudie comme une variation de l'identité ;

— $j_\Delta = |B_C|$. Le nombre de valeurs explicites de j est donc fonction de $|B_C|$ et est donc "petit".

La matrice j est construite pour chaque opération de passage de l'étape de construction t à $t+1$. Observons que pour éviter de dépendre du nombre de cellules de la structure topologique à l'étape t, **sa construction ne doit pas dépendre de ses $|B_{C'}|$ lignes ou de ses $|B_{C''}|$colonnes**.

Remarque. Le nombre de valeurs explicites j_Δ suppose que l'identité de j est représentée implicitement, et que j est représentée sous forme de matrice creuse, ce qui induit deux prérequis d'implémentation. On retrouve ces prérequis pour toutes les matrices de l'analyse étudiées comme variation de l'identité. Pour simplifier, ces prérequis sont mentionnés sans être motivés par la suite.

Prérequis d'implémentation.

— Représentation en matrice creuse : les valeurs nulles sont implicites ;

— Représentation implicite de l'identité : $\phi[x, x] = 1$ est implicite ;

— Construction d'une matrice qui ne dépend pas de son nombre de lignes ni de son nombre de colonnes.

Matrice s. Rappelons que pour toute cellule σ de $B_{C''}$, $\sigma s = \sigma$. Dit autrement, s **est l'inclusion de** $B_{C''}$ **dans** $B_{C'}$. En conséquence, il n'y a aucune valeur explicite. Les caractéristiques matricielles de s sont :

— $|B_{C''}|$ lignes et $|B_{C'}|$ colonnes ;

— on l'étudie comme une variation de l'identité ;

— $s_\Delta = 0$;

La matrice s est construite pour chaque opération de passage de l'étape de construction t à $t+1$. Observons que pour éviter de dépendre du nombre de cellules de la structure topologique

à l'étape t, **sa construction ne doit pas dépendre de ses $|B_{C''}|$ lignes ni de ses $|B_{C'}|$ colonnes**.

Prérequis d'implémentation.

- Représentation en matrice creuse : les valeurs nulles sont implicites ;
- Représentation implicite de l'identité : $\phi[x, x] = 1$ est implicite ;
- Construction d'une matrice qui ne dépend pas de son nombre de lignes ni de son nombre de colonnes.

2.1.2-b Analyse du calcul de l'équivalence homologique γ.

L'équivalence homologique γ est calculée pour chaque opération de passage de l'étape de construction t à $t + 1$.

Complexe de chaînes (C, ∂). Rappelons que C est engendré par les cellules non-survivantes de $B_{C'}$, et que $|B_C| = |B_{C'}| - |B_{C''}|$. De plus, la matrice ∂ est calculée à partir de la formule $\partial = i\partial' r$. Analysons ce calcul :

- elle a $|B_C|$ lignes et $|B_C|$ colonnes. D'après l'hypothèse (ii) du paragraphe 2.1.1-a, ses dimensions sont "petites" ;
- on l'étudie comme une variation de la matrice nulle ;
- il n'est pas possible d'étudier précisément ∂_Δ dans le cas général car nous n'avons pas d'hypothèse sur ∂'. Cependant, notons que dans le pire des cas $\partial_\Delta = |B_C|^2$, et est donc "petit" car il est fonction $|B_C|$.

Observons que pour optimiser le calcul de ∂ :

- l'ordre du produit $i\partial' r$ n'importe pas car il existe dans tous les cas une dimension exploitable (cf paragraphe 2.1.1-d) :
  - dans le cas $i(\partial' r)$, les $|B_C|$ colonnes de r sont exploitables pour calculer $(\partial' r)$ et $i(\partial' r)$;
  - dans le cas $(i\partial')r$, les $|B_C|$ lignes de i sont exploitables pour calculer $(i\partial')$ et $(i\partial')r$.
- la matrice ∂' est critique dans ce produit du fait de ses dimensions, que l'on choisisse de faire $(i\partial')r$ ou $i(\partial' r)$. Plus précisément, pour optimiser **le produit avec ∂', il doit être fait en parcourant** :
  - soit les $|B_C|$ colonnes de r ;
  - soit les $|B_C|$ lignes de i.
- **la matrice ∂ est construite** à chaque opération de passage de l'étape de construction t à $t + 1$. Ses caractéristiques matricielles font que sa construction ne nécessite pas de prérequis d'implémentation.

Remarque. En général, le produit $i(\partial' r)$ est très légèrement plus efficace car r a au plus 1 valeur explicite par colonne là où i a 2 valeurs explicites par ligne. Le temps de calcul du produit dépend ensuite entièrement de la répartition des valeurs explicites dans la matrice ∂'. Le produit $i(\partial' r)$ dépend du plus grand nombre de cellules dans l'étoile directe d'une cellule, là où le produit $(i\partial')r$ dépend du plus grand nombre de cellules dans le bord d'une cellule.

Prérequis d'implémentation.

— Représentation en matrice creuse : les valeurs nulles sont implicites ;
— Produit matriciel $A \times B$ en parcourant les lignes de A si le produit est $(i\partial')r$;
Produit matriciel $A \times B$ en parcourant les colonnes de B si le produit est $i(\partial' r)$.
— Construire une matrice.

Réductions ρ et ρ^S. À ce stade, nous n'avons émis aucune hypothèse concernant les réductions ρ et ρ^S, qui peuvent être calculées de plusieurs façons. Nous supposons qu'à chaque étape :

— la réduction ρ est une réduction identité [5] (cf. définition 1.4.4). En conséquence :
  · $(C^B, \partial^B) = (C, \partial)$;
  · f et g sont l'identité ;
  · h est la matrice nulle.

  Les matrices f, g et h étant complètement implicites, elles ne sont pas représentées. Elles n'induisent aucun prérequis d'implémentation ;
— la réduction ρ^S est calculée à partir de $(C^B, \partial^B) = (C, \partial)$. Il existe plusieurs façons de calculer cette réduction. En conséquence, il n'est pas possible d'étudier précisément les matrices de ces réductions. Cependant, par définition, on sait que le complexe de chaînes (C^S, ∂^S) est soit "plus petit", soit identique à $(C^B, \partial^B) = (C, \partial)$, d'où, $|B_{C^S}| \leq |B_C|$. Les caractéristiques des matrices $f^S : C^B \to C^S$, $g^S : C^S \to C^B$ et $h^S : C^B \to C^B$ sont donc fonction de $|B_C|$. Ces matrices sont explicites, et sont **construites** à chaque étape de construction. En particulier, f^S et g^S sont étudiées comme des variations de l'identité.

Prérequis d'implémentation.
— Représentation en matrice creuse : les valeurs nulles sont implicites ;
— Représentation implicite de l'identité : $\phi[x, x] = 1$ est implicite ;
— Construire une matrice.

2.1.2-c Récapitulatif des prérequis d'implémentation pour la SECE

On constate que deux calculs sont récurrents dans le calcul des matriccs i, j, r et s et de l'équivalence homologique γ : la **construction de matrices** et le **produit matriciel**. Ce paragraphe reprend l'ensemble des prérequis d'implémentation constatés : pour chacun d'eux, nous rappelons les matrices dont il émane.

Remarque. Sur support numérique, les noms de matrice de ce paragraphe sont des liens hypertextes. Ils redirigent vers l'analyse les concernant si l'on clique dessus.

Structure de donnée.

Structure-A | Représentation en matrice creuse : les valeurs nulles sont implicites. Ce prérequis est nécessaire pour le calcul de toutes les matrices manipulées.

Structure-B | Représentation implicite de l'identité : $\phi[x, x] = 1$ est implicite. Ce prérequis est nécessaire pour le calcul de : j, s, f^S, g^S.

5. Choisir une autre réduction pour ρ implique que (C^B, ∂^B) est plus "gros" que (C, ∂) en nombre de générateurs ($|B_{C^B}| > |B_C|$). Cela revient à augmenter inutilement la taille des données manipulées.

Construction de matrices.

Construction-A | Construire une matrice. Ce prérequis est nécessaire pour le calcul de : ∂, f^S, g^S, h^S, ∂^S.

Construction-B | Construction d'une matrice qui ne dépend pas de son nombre de colonnes. Ce prérequis est nécessaire pour le calcul de : i.

Construction-C | Construction d'une matrice qui ne dépend pas de son nombre de lignes ni de son nombre de colonnes. Ce prérequis est nécessaire pour le calcul de : j,s.

Construction-D | Construction d'une matrice qui ne dépend pas de son nombre de lignes. Ce prérequis est nécessaire pour le calcul de : r.

Produit impliquant une matrice.

Produit-A | Produit matriciel $A \times B$ en parcourant les lignes de A. Ce prérequis peut-être nécessaire pour le calcul de ∂, selon le produit matriciel choisi.

Produit-B | Produit matriciel $A \times B$ en parcourant les colonnes de B. Ce prérequis peut-être nécessaire pour le calcul de ∂, selon le produit matriciel choisi.

2.1.3 Identification

Dans ce paragraphe, nous considérons une SECE

$$(C, \partial) \underset{\mathsf{r}}{\overset{\mathsf{i}}{\rightleftarrows}} (C^t, \partial^t) \underset{\mathsf{s}}{\overset{\mathsf{j}}{\rightleftarrows}} (C^{t+1}, \partial^{t+1})$$

induite par une identification, et l'équivalence homologique

$$\gamma^t : (C^t, \partial^t) \overset{\rho^t}{\Longleftarrow} (C^{B,t}, \partial^{B,t}) \overset{\rho^{S,t}}{\Longrightarrow} (C^{S,t}, \partial^{S,t})$$

Nous analysons le calcul de l'équivalence homologique γ^{t+1} par application du théorème SECE, et par modification de γ^t sans émettre d'hypothèses sur les matrices ∂^t, $\partial^{B,t}$, $\partial^{S,t}$ ou sur les réductions ρ, ρ^S, ρ^t et $\rho^{S,t}$. L'objectif de ce paragraphe est d'établir l'ensemble des prérequis d'implémentation qui doivent être satisfaits pour optimiser le calcul de l'équivalence homologique γ^{t+1} lorsque le passage de l'étape t à l'étape $t+1$ se fait par identification. Optimiser ce calcul revient à profiter du "petit" nombre de cellules impactées par l'identification (cf. hypothèse (ii) du paragraphe 2.1.1-a). Nous distinguons deux cas d'étude :

— le cas où l'on peut se passer du calcul des générateurs d'homologie et où le processus de construction étudié n'est composé que d'identifications. Dans ce cas, seule une partie des matrices peut être calculée ;

— le cas où l'on souhaite pouvoir calculer les générateurs d'homologie et où le processus de construction étudié est composé d'au moins une désidentification. Dans ce cas, toutes les matrices doivent être calculées soit

  · pour exprimer les générateurs d'homologie du complexe de chaînes $(C^{S,t}, \partial^{S,t})$ dans (C^t, ∂^t) ;

  · parce qu'elles sont utilisées dans l'application du théorème SECE à la désidentification.

Remarque. L'étude qui suit est systématique, il est donc normal que certains mécanismes se retrouvent dans le calcul de plusieurs matrices. Par ailleurs, rappelons que γ est l'équivalence homologique induite par l'opération (cf. section 1.5), et dont les éléments sont "petits" du fait de l'hypothèse (ii).

2.1.3-a Sans les générateurs d'homologie

Lorsque l'on ne souhaite pas calculer les générateurs des groupes d'homologie, il est suffisant de calculer les matrices : $i^B, i^S, \partial^{B,t+1}, \partial^{S,t+1}, g^{t+1}, f^{S,t+1}$ et ∂^{t+1}. En particulier, il n'est pas nécessaire de pouvoir "exprimer" une chaîne de $(C^{S,t+1}, \partial^{S,t+1})$ dans $(C^{B,t+1}, \partial^{B,t+1})$ ou $(C^{t+1}, \partial^{t+1})$. La figure 2.3 reprend les différents éléments manipulés dans le calcul de γ^{t+1}. Le tableau 2.1 récapitule les caractéristiques des matrices connues au début de ce calcul, c'est-à-dire lorsque l'on connaît γ, γ^t et la SECE symbolisant l'identification de passage de l'étape t à l'étape $t+1$ du processus de construction. La figure 2.4 reprend les différentes formules du calcul de γ^{t+1} et illustre les dépendances qu'il existe entre les matrices manipulées. Analysons ces matrices.

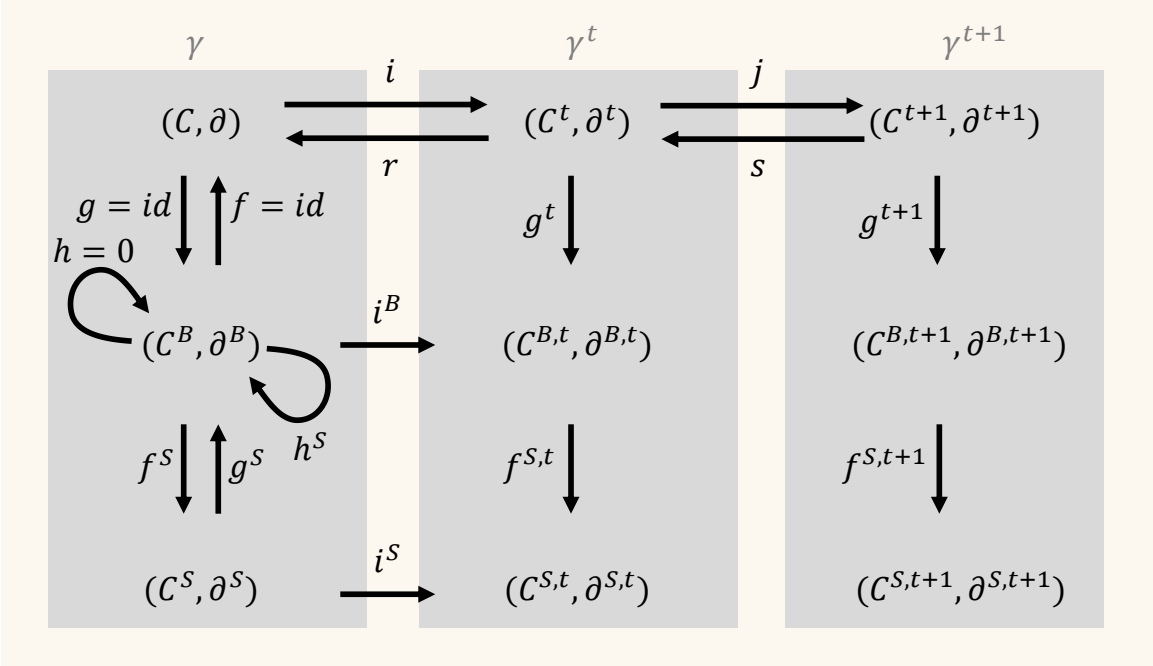


FIGURE 2.3 – Représentation des éléments manipulés dans l'application du théorème SECE à l'identification, lorsque les générateurs d'homologie ne sont pas calculés. Les équivalences homologiques sont mises en évidence via un fond gris.

Matrice i^B. Rappelons que $i^B = fig^t$ et que f est toujours l'identité (cf. paragraphe 2.1.2-b). On peut donc simplifier ce produit en $i^B = ig^t$. Ses caractéristiques matricielles sont :

- $|B_{C^B}|$ lignes et $|B_{C^{B,t}}|$ colonnes. Or, $|B_{C^B}| = |B_C|$ et $|B_C|$ est "petit" par rapport à $|B_{C^{B,t}}|$, d'après l'hypothèse (ii) du paragraphe 2.1.1-a ;
- on l'étudie comme une variation de la matrice nulle ;
- il n'est pas possible d'étudier précisément i^B_Δ dans le cas général car nous n'avons pas d'hypothèse sur g^t. Pour autant, notons que ig^t est une restriction de g^t aux cellules dans l'image de i, c'est-à-dire aux cellules non-survivantes, et aux cellules survivantes identifiées avec au moins une cellule non-survivante. Sans hypothèses sur g^t, l'image par g^t de ces cellules est quelconque. Dans le pire des cas, pour p prenant ses valeurs parmi les dimensions de ces cellules, g^t peut associer une de ces p-cellules à toutes les p-cellules de $B_{C^{B,t}}$. Dans le meilleur des cas, g^t est une association un-à-un, auquel cas $i^B_\Delta = 2|B_C|$.

M	Nb. de lignes	Nb. de colonnes	Variation	M_Δ
		$SECE$		
i	$\lvert B_C\rvert$	$\lvert B_{C^t}\rvert$	$[0]$	$2\lvert B_C\rvert$
j	$\lvert B_{C^t}\rvert$	$\lvert B_{C^{t+1}}\rvert$	$[1]$	$\lvert B_C\rvert$
r	$\lvert B_{C^t}\rvert$	$\lvert B_C\rvert$	$[0]$	$\lvert B_C\rvert$
s	$\lvert B_{C^{t+1}}\rvert$	$\lvert B_{C^t}\rvert$	$[1]$	0
		γ		
∂	$\lvert B_C\rvert$	$\lvert B_C\rvert$	$[0]$	$\Lambda \leq \lvert B_C\rvert^2$
f	$\lvert B_{C^B}\rvert = \lvert B_C\rvert$	$\lvert B_C\rvert$	$[1]$	0
g	$\lvert B_C\rvert$	$\lvert B_{C^B}\rvert = \lvert B^C\rvert$	$[1]$	0
h	$\lvert B_{C^B}\rvert = \lvert B_C\rvert$	$\lvert B_{C^B}\rvert = \lvert B_C\rvert$	$[0]$	0
∂^B	$\lvert B_{C^B}\rvert = \lvert B_C\rvert$	$\lvert B_{C^B}\rvert = \lvert B_C\rvert$	$[0]$	$\Lambda \leq \lvert B_C\rvert^2$
f^S	$\lvert B_{C^B}\rvert = \lvert B_C\rvert$	$\lvert B_{C^S}\rvert$	$[1]$	$\Lambda \leq \lvert B_C\rvert \times \lvert B_{C^S}\rvert$
g^S	$\lvert B_{C^S}\rvert$	$\lvert B_{C^B}\rvert = \lvert B_C\rvert$	$[1]$	$\Lambda \leq \lvert B_{C^S}\rvert \times \lvert B_C\rvert$
h^S	$\lvert B_{C^B}\rvert = \lvert B_C\rvert$	$\lvert B_{C^B}\rvert = \lvert B_C\rvert$	$[0]$	$\Lambda \leq \lvert B_C\rvert^2$
∂^S	$\lvert B_{C^S}\rvert$	$\lvert B_{C^S}\rvert$	$[0]$	$\Lambda \leq \lvert B_{C^S}\rvert^2$
		γ^t		
∂^t	$\lvert B_{C^t}\rvert$	$\lvert B_{C^t}\rvert$	$[0]$	$\Lambda \leq \lvert B_{C^t}\rvert^2$
g^t	$\lvert B_{C^t}\rvert$	$\lvert B_{C^{B,t}}\rvert$	$[1]$	$\Lambda \leq \lvert B_{C^t}\rvert \times \lvert B_{C^{B,t}}\rvert$
$\partial^{B,t}$	$\lvert B_{C^{B,t}}\rvert$	$\lvert B_{C^{B,t}}\rvert$	$[0]$	$\Lambda \leq \lvert B_{C^{B,t}}\rvert^2$
$f^{S,t}$	$\lvert B_{C^{B,t}}\rvert$	$\lvert B_{C^{S,t}}\rvert$	$[1]$	$\Lambda \leq \lvert B_{C^{B,t}}\rvert \times \lvert B_{C^{S,t}}\rvert$
$\partial^{S,t}$	$\lvert B_{C^{S,t}}\rvert$	$\lvert B_{C^{S,t}}\rvert$	$[0]$	$\Lambda \leq \lvert B_{C^{S,t}}\rvert^2$

TABLE 2.1 – Les caractéristiques matricielles connues au début du calcul de γ^{t+1}, lorsque les générateurs d'homologie ne sont pas calculés. La colonne "Variation" indique qu'une matrice M est étudiée comme une variation de la matrice nulle ($[0]$) ou de l'identité ($[1]$). La colonne "M_Δ" indique son nombre de valeurs explicites. Le symbole Λ indique qu'une donnée ne peut pas être étudiée précisément faute d'hypothèses la concernant.

Observons que :

— **le produit ig^t se fait en parcourant** les $\lvert B_C\rvert$ lignes de i, les dimensions de g^t n'étant pas exploitables pour le produit (cf. paragraphe 2.1.1-d) ;

— la matrice qui résulte de ce produit est **construite à chaque opération de passage de l'étape de construction t à $t+1$, sans dépendre de ses $\lvert B_{C^{B,t}}\rvert$ colonnes**.

Prérequis d'implémentation.

— Représentation en matrice creuse : les valeurs nulles sont implicites ;

— Produit matriciel $A \times B$ en parcourant les lignes de A

— Construction d'une matrice qui ne dépend pas de son nombre de colonnes.

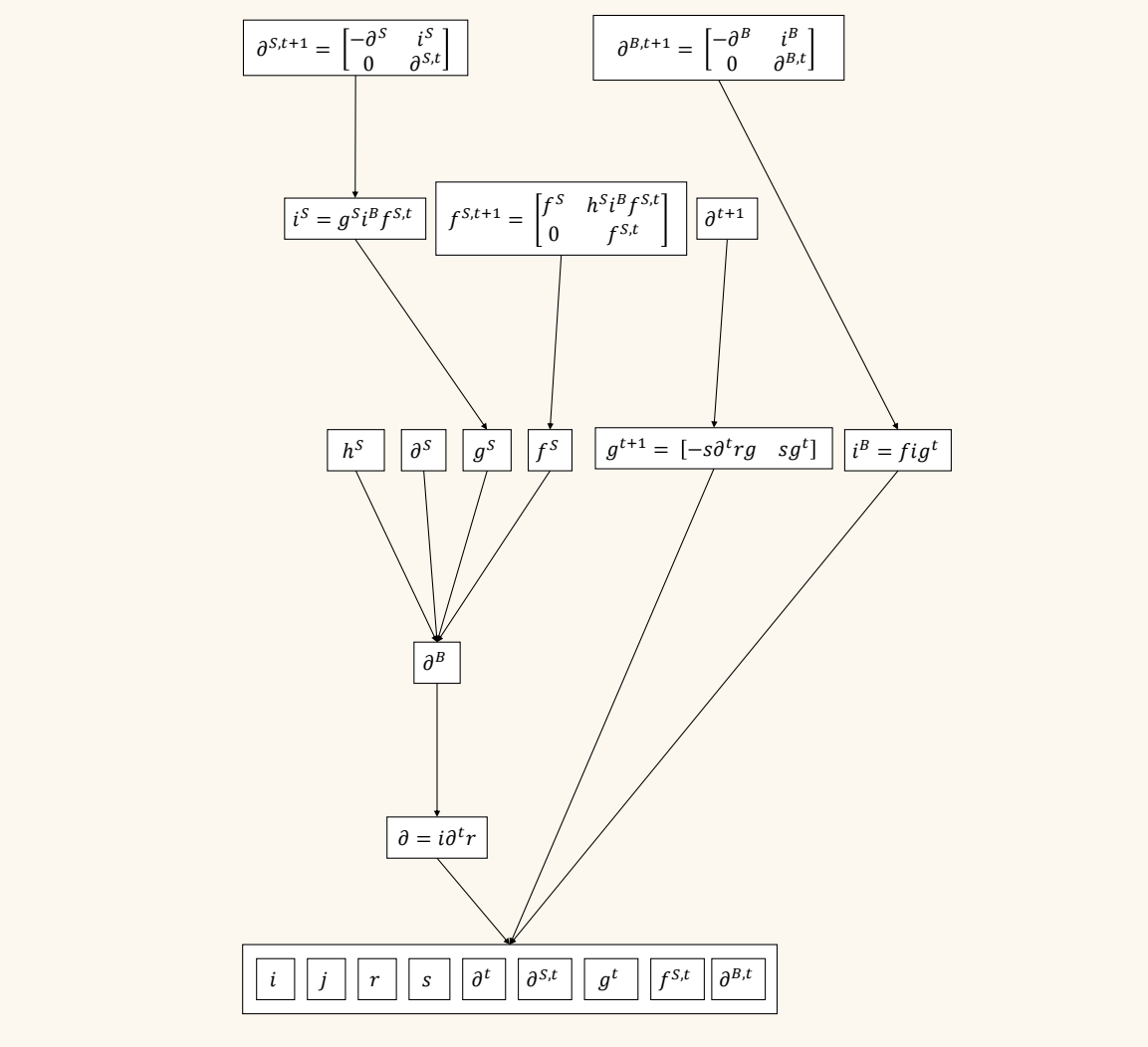


FIGURE 2.4 – Graphe de dépendance entre les matrices manipulées lors de l'application du théorème SECE à l'identification, et lorsque les générateurs d'homologie ne sont pas calculés. Une flèche d'un nœud A vers un nœud B indique que A a besoin de B pour son calcul. Pour simplifier le schéma, toutes les relations ne sont pas représentées. Les matrices $f = id$, $g = id$ et $h = 0$ ne sont pas représentées.

Matrice i^S. Rappelons que $i^S = g^S i^B f^{S,t}$. Ses caractéristiques matricielles sont :

- $|B_{C^S}|$ lignes et $|B_{C^{S,t}}|$ colonnes. Notons que $|B_{C^S}|$est "petit" d'après l'hypothèse (ii) du paragraphe 2.1.1-a ;
- on l'étudie comme une variation de la matrice nulle ;
- il n'est pas possible d'étudier précisément i^S_Δ dans le cas général car nous n'avons pas d'hypothèse sur $f^{S,t}$.

Observons que :

- l'ordre du produit $g^S i^B f^{S,t}$ n'importe pas car il existe dans tous les cas une dimension exploitable (cf paragraphe 2.1.1-d) :

  - dans le cas $g^S(i^B f^{S,t})$, les $|B_{C^B}|$ lignes de i^B sont exploitables pour calculer $(i^B f^{S,t})$, et les $|B_{C^S}|$ lignes de g^S sont exploitables pour calculer $g^S(i^B f^{S,t})$;
  - dans le cas $(g^S i^B) f^{S,t}$, les $|B_{C^S}|$ lignes de g^S sont exploitables pour calculer $(g^S i^B)$ et $(g^S i^B) f^{S,t}$.

— la matrice qui résulte de ce produit est **construite à chaque opération de passage de l'étape t à l'étape $t+1$, sans dépendre de ses $|B_{C^{S,t}}|$ colonnes.**

Prérequis d'implémentation.

— Représentation en matrice creuse : les valeurs nulles sont implicites ;
— Produit matriciel $A \times B$ en parcourant les lignes de A
— Construction d'une matrice qui ne dépend pas de son nombre de colonnes.

Matrice $\partial^{B,t+1}$. Rappelons que $\partial^{B,t+1} = \begin{pmatrix} -\partial^B & i^B \\ 0 & \partial^{B,t} \end{pmatrix}$. En particulier, cette matrice est calculée comme une modification de la matrice $\partial^{B,t}$. Ses caractéristiques matricielles sont :

— $|B_{C^B}| + |B_{C^{B,t}}|$ lignes et $|B_{C^B}| + |B_{C^{B,t}}|$ colonnes. Or, $|B_{C^B}|$ est "petit" par rapport à $|B_{C^{B,t}}|$, d'après l'hypothèse (ii) du paragraphe 2.1.1-a. Dit autrement, le nombre de lignes et colonnes ajoutées à l'issue de la modification de $\partial^{B,t}$ est "petit" ;
— on l'étudie comme une variation de la matrice nulle ;
— $\partial_\Delta^{B,t+1} = \partial_\Delta^{B,t} + i_\Delta^B + \partial_\Delta^B$. Dit autrement, le calcul de $\partial^{B,t+1}$ par modification de $\partial^{B,t}$ implique l'ajout de i_Δ^B et ∂_Δ^B valeurs explicites dans $\partial^{B,t}$.

Observons que :

— ∂^B **est multipliée par** -1 ;
— calculer $\partial^{B,t+1}$ revient à **fusionner** 4 **matrices** (cf. paragraphe 2.1.1-e), dont l'une est une matrice nulle. En particulier, cette fusion se fait :
  - **par modification de la matrice en bas à droite**, à savoir $\partial^{B,t}$;
  - **en parcourant les $|B_{C^B}|$ lignes de $-\partial^B$ et i^B**, pour éviter une dépendance aux dimensions de $\partial^{B,t}$, et donc indirectement au nombre de cellules de la structure topologique.

Prérequis d'implémentation.

— Représentation en matrice creuse : les valeurs nulles sont implicites ;
— Produit d'une matrice avec un scalaire ;
— Fusion de matrices $D = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$

Matrice $\partial^{S,t+1}$. Rappelons que $\partial^{S,t+1} = \begin{pmatrix} -\partial^S & i^S \\ 0 & \partial^{S,t} \end{pmatrix}$. En particulier, cette matrice est calculée comme une modification de $\partial^{S,t}$. Ses caractéristiques matricielles sont :

— $|B_{C^S}| + |B_{C^{S,t}}|$ lignes et $|B_{C^S}| + |B_{C^{S,t}}|$ colonnes. Or, $|B_{C^S}|$ est "petit" par rapport à $|B_{C^{S,t}}|$, d'après l'hypothèse (ii) du paragraphe 2.1.1-a. Dit autrement, le nombre de lignes et colonnes ajoutées à l'issue de la modification de $\partial^{S,t}$ est "petit" ;

— on l'étudie comme une variation de la matrice nulle ;

— $\partial_\Delta^{S,t+1} = \partial_\Delta^{S,t} + i_\Delta^S + \partial_\Delta^S$. Dit autrement, le calcul de $\partial^{S,t+1}$ par modification de $\partial^{S,t}$ implique l'ajout de i_Δ^S et ∂_Δ^S valeurs explicites dans $\partial^{S,t}$.

Observons que :

— ∂^S **est multipliée par** -1 ;

— calculer $\partial^{S,t+1}$ revient à **fusionner** 4 **matrices** (cf. paragraphe 2.1.1-e), dont l'une est une matrice nulle. En particulier, cette fusion se fait :

· **par modification de la matrice en bas à droite**, à savoir $\partial^{S,t}$;

· **en parcourant les** $|B_{C^S}|$ **lignes de** $-\partial^S$ **et** i^S pour éviter une dépendance aux dimensions de $\partial^{S,t}$, et donc indirectement au nombre de cellules de la structure topologique.

Prérequis d'implémentation.

— Représentation en matrice creuse : les valeurs nulles sont implicites ;

— Produit d'une matrice avec un scalaire ;

— Fusion de matrices $D = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$.

Matrice g^{t+1}**.** Rappelons que $g^{t+1} = \begin{pmatrix} -s\partial^t r g & sg^t \end{pmatrix}$ et que g est toujours l'identité (cf. paragraphe 2.1.2-b). Ce calcul se simplifie en $\begin{pmatrix} -s\partial^t r & sg^t \end{pmatrix}$. En particulier, g^{t+1} est calculée comme une modification de la matrice g^t à l'issue de laquelle ses caractéristiques matricielles sont :

— $|B_{C^t}| - |B_C|$ lignes, et $|B_{C^B}| + |B_{C^{B,t}}|$ colonnes. Or, $|B_C|$ est "petit" par rapport à $|B_{C^t}|$, d'après l'hypothèse (ii) du paragraphe 2.1.1-a, de même que $|B_{C^B}|$ qui est "petit" par rapport à $|B_{C^{B,t}}|$. Dit autrement, le nombre de lignes supprimées et le nombre de colonnes ajoutées à l'issue de la modification de g^t est "petit" ;

— on l'étudie comme une variation de l'identité ;

— il n'est pas possible d'étudier précisément g_Δ^{t+1} dans le cas général car nous n'avons pas d'hypothèses sur ∂^t et g^t.

Observons que :

— pour le produit $-s\partial^t r$,

· l'ordre du produit importe et doit être $-s(\partial^t r)$ de manière à **effectuer ce produit matriciel en exploitant le fait que** r **n'ait que** $|B_C|$ **colonnes**. Faire $-(s\partial^t)r$ pose problème car aucune des dimensions des matrices de $s\partial^t$ n'est exploitable ;

· la matrice qui résulte de ce produit **est multipliée par** -1 ;

· la matrice qui résulte de ce produit doit être **construite sans dépendre de ses** $|B_{C^t}| - |B_C|$ **lignes**.

— pour le produit sg^t,

· aucune des dimensions des deux matrices n'est exploitable. Ce calcul ne peut pas être fait sous la forme d'un produit matriciel ;

  - s représente l'inclusion de C^{t+1} dans C^t. Dit autrement, seules les cellules survivantes de C^t ont un antécédent par s dans C^{t+1} ;
  - ce calcul revient à **supprimer les lignes de g^t qui représentent une cellule non survivante de C^t**.
- calculer g^{t+1} implique de fusionner 2 matrices (cf. paragraphe 2.1.1-e), à savoir $-s\partial^t r$ et sg^t. Pour ce faire, on commence par calculer sg^t, puis on fusionne $-s\partial^t r$.

Prérequis d'implémentation.

- Représentation en matrice creuse : les valeurs nulles sont implicites ;
- Représentation implicite de l'identité : $\phi[x, x] = 1$ est implicite ;
- Produit matriciel $A \times B$ en parcourant les colonnes de B ;
- Produit d'une matrice avec un scalaire, sans dépendre de son nombre de lignes ;
- Construction d'une matrice qui ne dépend pas de son nombre de lignes ;
- Suppression d'une ligne d'une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes ;
- Fusion de matrices $B = \begin{bmatrix} A & B \end{bmatrix}$.

Matrice $f^{S,t+1}$. Rappelons que $f^{S,t+1} = \begin{pmatrix} f^S & h^S i^B f^{S,t} \\ 0 & f^{S,t} \end{pmatrix}$. En particulier, cette matrice est calculée comme une modification de $f^{S,t}$. Ses caractéristiques matricielles sont :

- $|B_{C^B}| + |B_{C^{B,t}}|$ lignes et $|B_{C^S}| + |B_{C^{S,t}}|$ colonnes. Or, $|B_{C^B}|$ est "petit" par rapport à $|B_{C^{B,t}}|$, d'après l'hypothèse (ii) du paragraphe 2.1.1-a, de même que $|B_{C^S}|$ qui est "petit" par rapport à $|B_{C^{S,t}}|$. Dit autrement, le nombre de lignes et colonnes ajoutées à l'issue de la modification de $f^{S,t}$ est "petit" ;
- on l'étudie comme une variation de l'identité ;
- il n'est pas possible d'étudier précisément $f_\Delta^{S,t+1}$ dans le cas général car nous n'avons pas d'hypothèse sur $h^S i^B f^{S,t}$.

Observons que :

- pour le produit $h^S i^B f^{S,t}$,
  - l'ordre n'importe pas car il existe dans tous les cas une dimension exploitable (cf paragraphe 2.1.1-d) :
    - ▷ dans le cas $h^S(i^B f^{S,t})$, les $|B_{C^B}|$ lignes de i^B sont exploitables pour calculer $(i^B f^{S,t})$, et les $|B_{C^B}|$ lignes de h^S sont exploitables pour calculer $h^S(i^B f^{S,t})$;
    - ▷ dans le cas $(h^S i^B) f^{S,t}$, les $|B_{C^S}|$ lignes de h^S sont exploitables pour calculer $(h^S i^B) f^{S,t}$;
  - la matrice qui résulte de ce produit **doit être construite sans dépendre de ses $|B_{C^{S,t}}|$ colonnes.**
- calculer $f^{S,t+1}$ revient à **fusionner 4 matrices** (cf. paragraphe 2.1.1-e), dont l'une est une matrice nulle. En particulier, cette fusion se fait :
  - par **modification de la matrice en bas à droite**, à savoir $f^{S,t}$;
  - **sans parcourir les $|B_{C^{S,t}}|$ colonnes de** $h^S i^B f^{S,t}$ pour éviter une dépendance à $|B_{C^{S,t}}| \leq |B_{C^{B,t}}| \geq |B_{C^t}|$, et donc au nombre de cellules de la structure topologique.

Prérequis d'implémentation.

- Représentation en matrice creuse : les valeurs nulles sont implicites ;
- Représentation implicite de l'identité : $\phi[x, x] = 1$ est implicite ;
- Produit matriciel $A \times B$ en parcourant les lignes de A
- Construction d'une matrice qui ne dépend pas de son nombre de colonnes
- Fusion de matrices $D = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$.

Matrice ∂^{t+1}. Cette matrice représente le bord du complexe de chaînes associé à la structure topologique à l'étape $t + 1$, c'est-à-dire après que l'identification ait été appliquée. En conséquence, elle peut être calculée comme une modification de ∂^t dans laquelle, pour chaque p-cellule non-survivante σ' de[6] (C^t, ∂^t) identifiée avec une p-cellule survivante σ :

- on **somme les colonnes σ' et σ**. Intuitivement, l'ajout de colonne représente la modification du bord des $(p + 1)$-cellules dont le bord est composé de σ' ;
- on **supprime la ligne** σ' et on **supprime la colonne** σ' car σ' n'existe plus après identification.

Remarque. La matrice ∂^t est utilisée dans le calcul de g^{t+1}, comme le montre la figure 2.4. En conséquence, ∂^{t+1} doit être calculée après g^{t+1}.

Prérequis d'implémentation.

- Représentation en matrice creuse : les valeurs nulles sont implicites ;
- Suppression d'une ligne d'une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes ;
- Somme de deux colonnes dans une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes ;
- Suppression d'une colonne d'une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes.

Exemple. Cet exemple illustre les modifications induites par une identification au niveau du complexe de chaînes (C^t, ∂^t) de la figure 2.5, associé à l'ensemble semi-simplicial (K, d) de la figure 1.8. Pour rappel, cette identification est caractérisée par $I = (I^p)^{0 \leq p \leq 2}$, telle que :

$$I^0 = \{\{v_0\}, \{v_1, v_3\}, \{v_2, v_5\}, \{v_4\}\}$$
$$I^1 = \{\{e_0, e_5\}, \{e_1\}, \{e_2\}, \{e_3\}, \{e_4\}\}$$
$$I^2 = \{\{f_0\}, \{f_1\}\}$$

6. Plus précisément, σ' est un générateur de C^t engendré par une cellule de la structure topologique à l'étape t. Il en est de même pour σ.

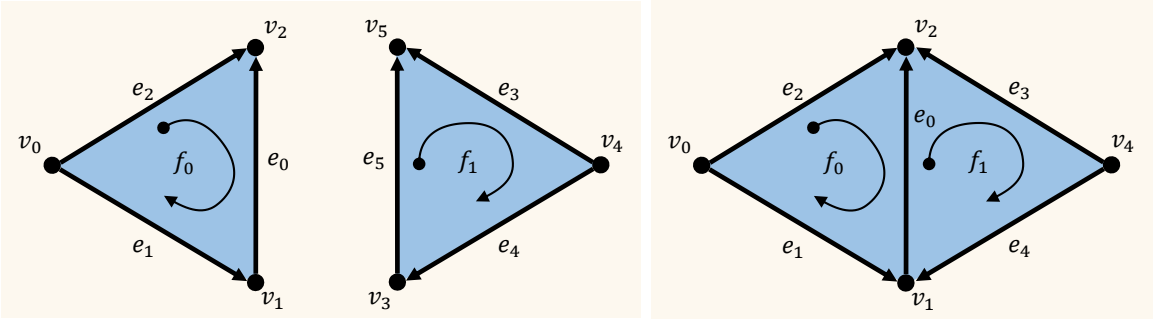


FIGURE 2.5 – Représentation graphique des complexes de chaînes (C^t, ∂^t) et $(C^{t+1}, \partial^{t+1})$ associés à un ensemble semi-simplicial sur lequel est appliquée une identification. (a) (C^t, ∂^t). (b) $(C^{t+1}, \partial^{t+1})$.

et $v_0\zeta = v_0$, $v_1\zeta = v_3\zeta = v_1$, $v_2\zeta = v_5\zeta = v_2$, $e_0\zeta = e_5\zeta = e_0$. La[7] matrice de bord ∂^t est :

$$
\begin{array}{c}
\partial^t \\
v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ e_0 \\ e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ f_0 \\ f_1
\end{array}
\begin{array}{c}
\begin{array}{cccccccccccccc}
v_0 & v_1 & v_2 & v_3 & v_4 & v_5 & e_0 & e_1 & e_2 & e_3 & e_4 & e_5 & f_0 & f_1
\end{array} \\
\left[\begin{array}{cccccccccccccc}
. & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & -1 & 1 & . & . & . & . & . & . & . & . & . & . & . \\
-1 & 1 & . & . & . & . & . & . & . & . & . & . & . & . \\
-1 & . & 1 & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & -1 & 1 & . & . & . & . & . & . & . & . \\
. & . & . & 1 & -1 & . & . & . & . & . & . & . & . & . \\
. & . & . & -1 & . & 1 & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & -1 & -1 & 1 & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & -1 & 1 & 1 & . & .
\end{array}\right]
\end{array}
$$

Modifions ∂^t :

— en supprimant les lignes v_3, v_5 et e_5 ;

— en sommant les colonnes v_3 dans v_1, v_5 dans v_2 et e_5 dans e_0 ;

— en supprimant les colonnes v_3, v_5 et e_5.

7. Pour simplifier l'exemple, on regroupe les différentes matrices de bord $(\partial^p)^{0\leq p\leq 2}$ en une seule.

∂^{t+1}	v_0	$v_1(+v_3)$	$v_2(+v_5)$	v_4	$e_0(+e_5)$	e_1	e_2	e_3	e_4	f_0	f_1
v_0	.	.	.	.	.	.	.	.	.	.	.
v_1	.	.	.	.	.	.	.	.	.	.	.
v_2	.	.	.	.	.	.	.	.	.	.	.
v_4	.	.	.	.	.	.	.	.	.	.	.
e_0	.	-1	1	.	.	.	.	.	.	.	.
e_1	-1	1	.	.	.	.	.	.	.	.	.
e_2	-1	.	1	.	.	.	.	.	.	.	.
e_3	.	.	1	-1	.	.	.	.	.	.	.
e_4	.	1	.	-1	.	.	.	.	.	.	.
f_0	.	.	.	.	-1	-1	1	.	.	.	.
f_1	.	.	.	.	1	.	.	-1	1	.	.

La matrice obtenue est ∂^{t+1}.

2.1.3-b Avec les générateurs d'homologie

Lorsque les générateurs des groupes d'homologie sont calculés, il faut calculer des matrices en plus de celles déjà étudiées, à savoir : $f^{t+1}, h^{t+1}, g^{S,t+1}, h^{S,t+1}$. Intuitivement, elles permettent "d'exprimer" une chaîne de $(C^{S,t+1}, \partial^{S,t+1})$ dans $(C^{B,t+1}, \partial^{B,t+1})$ ou $(C^{t+1}, \partial^{t+1})$. La figure 2.6 reprend les différents éléments manipulés dans le calcul de γ^{t+1}. Le tableau 2.2 récapitule les caractéristiques des matrices connues au début de ce calcul, c'est-à-dire lorsque l'on connaît γ, γ^t et la SECE symbolisant l'identification de passage de l'étape t à l'étape $t+1$ du processus de construction. La figure 2.7 reprend les différentes formules du calcul de γ^{t+1} et illustre les dépendances qu'il existe entre les matrices manipulées. Analysons ces matrices.

Remarque. Pour rappel, ce cas d'étude est équivalent à l'étude de l'application du théorème SECE à l'identification dans un processus de construction composé d'identifications et de désidentifications, que les générateurs d'homologie soient calculables ou non.

Matrice f^{t+1}. Rappelons que $f^{t+1} = \begin{pmatrix} 0 \\ f^t j \end{pmatrix}$. En particulier, cette matrice est calculée comme une modification de f^t. Ses caractéristiques matricielles sont :

- $|B_{C^B}| + |B_{C^{B,t}}|$ lignes et $|B_{C^t}| - |B_C|$ colonnes. Or, $|B_{C^B}|$ est "petit" par rapport à $|B_{C^{B,t}}|$, d'après l'hypothèse (ii) du paragraphe 2.1.1-a, de même que $|B_C|$ qui est "petit" par rapport à $|B_{C^t}|$. Dit autrement, le nombre de lignes ajoutées et le nombre de colonnes supprimées à l'issue de la modification de f^t est "petit".
- on l'étudie comme une variation de l'identité ;
- il n'est pas possible d'étudier précisément f^{t+1}_Δ dans le cas général car nous n'avons pas d'hypothèse sur f^t. Pour autant, notons que $f^t j$ revient à modifier l'image par f^t des cellules qui ont dans leur image une cellule non-survivante, en la remplaçant par la cellule survivante avec laquelle elle est identifiée.

Observons que :

- pour le produit $f^t j$,
  - aucune des dimensions des deux matrices n'est exploitable. Ce calcul ne peut pas être fait sous la forme d'un produit matriciel ;

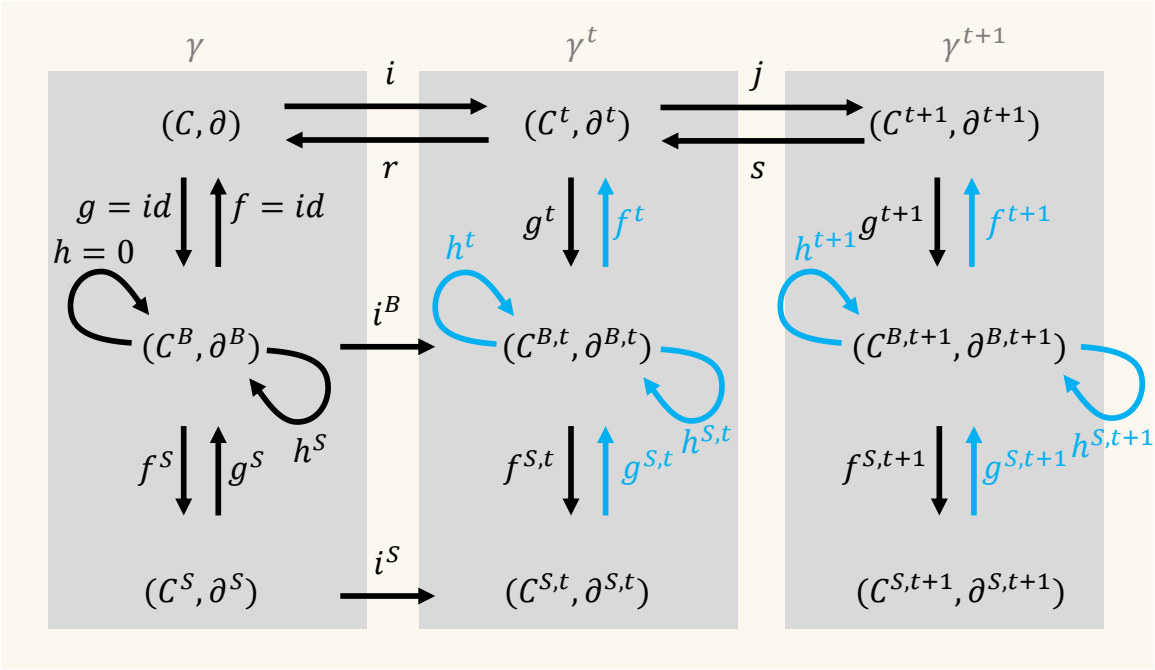


FIGURE 2.6 – Représentation des éléments impliqués dans le théorème SECE dans le cas de l'identification lorsque les générateurs d'homologie sont calculés. Les équivalences homologiques sont mises en évidence via un fond gris. En bleu, les éléments supplémentaires calculés lorsque les générateurs d'homologie le sont.

  - j représente l'identité pour toutes les cellules survivantes de C^t, et j associe les cellules non-survivantes de C^t aux cellules survivantes avec lesquelles ils sont identifiés ;
  - le produit $f^t j$ revient à **sommer les $|B_C|$ colonnes représentant des cellules non-survivantes de C^t avec les colonnes représentant les cellules survivantes** avec lesquelles elles sont identifiées, puis à **supprimer ces colonnes**.
- calculer f^{t+1} implique de fusionner 2 matrices (cf. paragraphe 2.1.1-e), dont l'une est une matrice nulle. **Cette opération revient à ajouter $|B_C|$ lignes vides dans $f^t j$.**

Prérequis d'implémentation.

- Représentation en matrice creuse : les valeurs nulles sont implicites ;
- Représentation implicite de l'identité : $\phi[x,x]=1$ est implicite ;
- Somme de deux colonnes dans une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes ;
- Suppression d'une colonne d'une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes ;
- Ajout de lignes vides dans une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes.

Matrice h^{t+1}. Rappelons que $h^{t+1} = \begin{pmatrix} -h & 0 \\ f^t r g & h^t \end{pmatrix}$, que g est toujours l'identité et que h est toujours la matrice nulle (cf. paragraphe 2.1.2-b). Cette formule peut donc se simplifier en

M	Nb. de lignes	Nb. de colonnes	Variation	M_Δ
$SECE$				
i	$\lvert B_C\rvert$	$\lvert B_{C^t}\rvert$	$[0]$	$2\lvert B_C\rvert$
j	$\lvert B_{C^t}\rvert$	$\lvert B_{C^{t+1}}\rvert$	$[1]$	$\lvert B_C\rvert$
r	$\lvert B_{C^t}\rvert$	$\lvert B_C\rvert$	$[0]$	$\lvert B_C\rvert$
s	$\lvert B_{C^{t+1}}\rvert$	$\lvert B_{C^t}\rvert$	$[1]$	0
γ				
∂	$\lvert B_C\rvert$	$\lvert B_C\rvert$	$[0]$	$\Lambda \leq \lvert B_C\rvert^2$
f	$\lvert B_{C^B}\rvert = \lvert B_C\rvert$	$\lvert B_C\rvert$	$[1]$	0
g	$\lvert B_C\rvert$	$\lvert B_{C^B}\rvert = \lvert B^C\rvert$	$[1]$	0
h	$\lvert B_{C^B}\rvert = \lvert B_C\rvert$	$\lvert B_{C^B}\rvert = \lvert B_C\rvert$	$[0]$	0
∂^B	$\lvert B_{C^B}\rvert = \lvert B_C\rvert$	$\lvert B_{C^B}\rvert = \lvert B_C\rvert$	$[0]$	$\Lambda \leq \lvert B_C\rvert^2$
f^S	$\lvert B_{C^B}\rvert = \lvert B_C\rvert$	$\lvert B_{C^S}\rvert$	$[1]$	$\Lambda \leq \lvert B_C\rvert \times \lvert B_{C^S}\rvert$
g^S	$\lvert B_{C^S}\rvert$	$\lvert B_{C^B}\rvert = \lvert B_C\rvert$	$[1]$	$\Lambda \leq \lvert B_{C^S}\rvert \times \lvert B_C\rvert$
h^S	$\lvert B_{C^B}\rvert = \lvert B_C\rvert$	$\lvert B_{C^B}\rvert = \lvert B_C\rvert$	$[0]$	$\Lambda \leq \lvert B_C\rvert^2$
∂^S	$\lvert B_{C^S}\rvert$	$\lvert B_{C^S}\rvert$	$[0]$	$\Lambda \leq \lvert B_{C^S}\rvert^2$
γ^t				
∂^t	$\lvert B_{C^t}\rvert$	$\lvert B_{C^t}\rvert$	$[0]$	$\Lambda \leq \lvert B_{C^t}\rvert^2$
f^t	$\lvert B_{C^{B,t}}\rvert$	$\lvert B_{C^t}\rvert$	$[1]$	$\Lambda \leq \lvert B_{C^{B,t}}\rvert \times \lvert B_{C^t}\rvert$
g^t	$\lvert B_{C^t}\rvert$	$\lvert B_{C^{B,t}}\rvert$	$[1]$	$\Lambda \leq \lvert B_{C^t}\rvert \times \lvert B_{C^{B,t}}\rvert$
h^t	$\lvert B_{C^{B,t}}\rvert$	$\lvert B_{C^{B,t}}\rvert$	$[0]$	$\Lambda \leq \lvert B_{C^{B,t}}\rvert^2$
$\partial^{B,t}$	$\lvert B_{C^{B,t}}\rvert$	$\lvert B_{C^{B,t}}\rvert$	$[0]$	$\Lambda \leq \lvert B_{C^{B,t}}\rvert^2$
$f^{S,t}$	$\lvert B_{C^{B,t}}\rvert$	$\lvert B_{C^{S,t}}\rvert$	$[1]$	$\Lambda \leq \lvert B_{C^{B,t}}\rvert \times \lvert B_{C^{S,t}}\rvert$
$g^{S,t}$	$\lvert B_{C^{S,t}}\rvert$	$\lvert B_{C^{B,t}}\rvert$	$[1]$	$\Lambda \leq \lvert B_{C^{S,t}}\rvert \times \lvert B_{C^{B,t}}\rvert$
$h^{S,t}$	$\lvert B_{C^{B,t}}\rvert$	$\lvert B_{C^{B,t}}\rvert$	$[0]$	$\Lambda \leq \lvert B_{C^{B,t}}\rvert^2$
$\partial^{S,t}$	$\lvert B_{C^{S,t}}\rvert$	$\lvert B_{C^{S,t}}\rvert$	$[0]$	$\Lambda \leq \lvert B_{C^{S,t}}\rvert^2$

TABLE 2.2 – Les caractéristiques matricielles connues au début du calcul de γ^{t+1} lorsque les générateurs d'homologie sont calculés. La colonne "Variation" indique qu'une matrice M est étudiée comme une variation de la matrice nulle ($[0]$) ou de l'identité ($[1]$). La colonne "M_Δ" indique son nombre de valeurs explicites. Le symbole Λ indique qu'une donnée n'est pas connue.

$h^{t+1} = \begin{pmatrix} 0 & 0 \\ f^t r & h^t \end{pmatrix}$. En particulier, cette matrice est calculée comme une modification de h^t. Ses caractéristiques matricielles sont :

- $\lvert B_{C^B}\rvert + \lvert B_{C^{B,t}}\rvert$ lignes et $\lvert B_{C^B}\rvert + \lvert B_{C^{B,t}}\rvert$ colonnes. Or, $\lvert B_{C^B}\rvert$ est "petit" par rapport à $\lvert B_{C^{B,t}}\rvert$, d'après l'hypothèse (ii) du paragraphe 2.1.1-a. Dit autrement, le nombre de lignes et de colonnes ajoutées à l'issue de la modification de h^t est "petit".
- on l'étudie comme une variation de la matrice nulle ;
- il n'est pas possible d'étudier précisément h_Δ^{t+1} dans le cas général car nous n'avons pas d'hypothèse sur f^t.

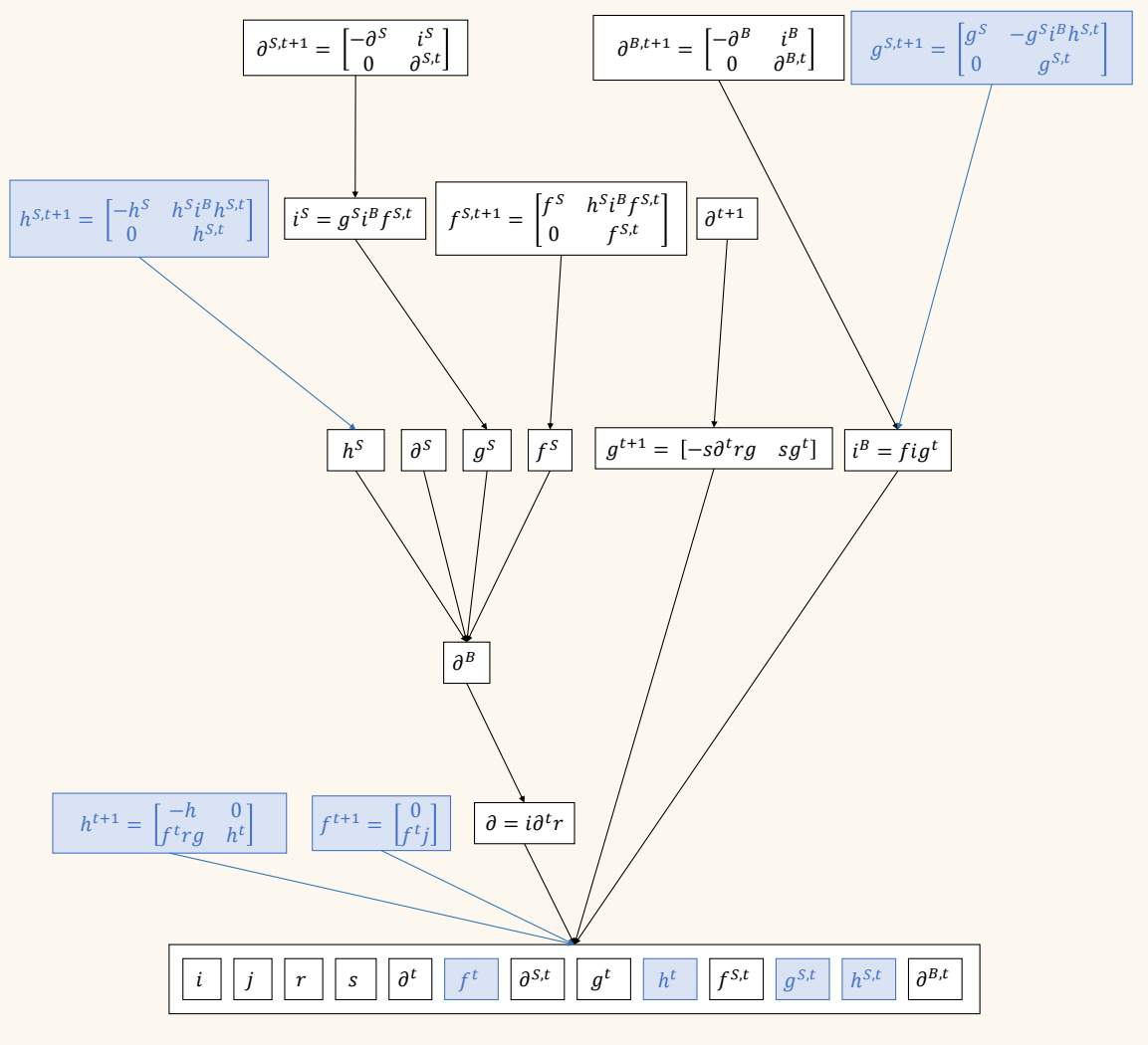


FIGURE 2.7 – Graphe de dépendance des matrices utilisées dans l'application du théorème SECE à l'identification, lorsque les générateurs d'homologie sont calculés. Toutes les relations ne sont pas représentées. Les matrices $f = id$, $g = id$ et $h = 0$ ne sont pas représentées. En bleu, les éléments supplémentaires calculés lorsque les générateurs d'homologie le sont.

Observons que :

- **le produit $f^t r$ se fait en parcourant** les $|B_C|$colonnes de r ;
- calculer h^{t+1} revient à fusionner 4 matrices (cf. paragraphe 2.1.1-e),
  - par **modification de la matrice en bas à droite**, à savoir h^t ;
  - sans parcourir les $|B_{C^{B,t}}|$ lignes de $f^t rg$ pour éviter une dépendance à $|B_{C^{B,t}}| \geq |B_{C^t}|$, et donc au nombre de cellules de la structure topologique.

Prérequis d'implémentation.

- Représentation en matrice creuse : les valeurs nulles sont implicites ;
- Produit matriciel $A \times B$ en parcourant les colonnes de B
- Fusion de matrices $D = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$.

Matrice $g^{S,t+1}$. Rappelons que $g^{S,t+1} = \begin{pmatrix} g^S & -g^S i^B h^{S,t} \\ 0 & g^{S,t} \end{pmatrix}$. En particulier, cette matrice est calculée comme une modification de $g^{S,t}$. Ses caractéristiques matricielles sont :

— $|B_{C^S}| + |B_{C^{S,t}}|$ lignes et $|B_{C^B}| + |B_{C^{B,t}}|$ colonnes. Or, $|B_{C^S}|$ est "petit" par rapport à $|B_{C^{S,t}}|$, d'après l'hypothèse (ii) du paragraphe 2.1.1-a, de même que $|B_{C^B}|$qui est "petit" par rapport à $|B_{C^{B,t}}|$. Dit autrement, le nombre de lignes et de colonnes ajoutées à l'issue de la modification de $g^{S,t}$ est "petit".

— on l'étudie comme une variation de l'identité ;

— il n'est pas possible d'étudier précisément $g_\Delta^{S,t+1}$ dans le cas général car nous n'avons pas d'hypothèse sur $h^{S,t}$.

Observons que :

— pour le produit $-g^S i^B h^{S,t}$,

· l'ordre n'importe pas car il existe dans tous les cas une dimension exploitable (cf paragraphe 2.1.1-d) ;

▷ dans le cas $-(g^S i^B) h^{S,t}$, les $|B_{C^S}|$ lignes de g^S sont exploitables pour calculer $(g^S i^B)$ et $-(g^S i^B) h^{S,t}$;

▷ dans le cas $-g^S (i^B h^{S,t})$, les $|B_{C^B}|$ lignes de i^B sont exploitables pour calculer $(i^B h^{S,t})$ et les $|B_{C^S}|$ lignes de g^S sont exploitables pour calculer $-g^S (i^B h^{S,t})$;

· **le produit $i^B h^{S,t}$ se fait en parcourant** les $|B_{C^B}|$lignes de i^B ;

· la matrice qui résulte du produit **est multipliée par -1**.

— calculer $g^{S,t+1}$ revient à fusionner 4 matrices (cf. paragraphe 2.1.1-e),

· par **modification de la matrice en bas à droite**, à savoir $g^{S,t}$;

· sans parcourir les $|B_{C^{B,t}}|$ colonnes de $-g^S i^B h^{S,t}$ pour éviter une dépendance à $|B_{C^{B,t}}| \geq |B_{C^t}|$, et donc au nombre de cellules de la structure topologique.

Prérequis d'implémentation.

— Représentation en matrice creuse : les valeurs nulles sont implicites ;

— Représentation implicite de l'identité : $\phi[x, x] = 1$ est implicite ;

— Produit d'une matrice avec un scalaire, sans dépendre de son nombre de colonnes ;

— Produit matriciel $A \times B$ en parcourant les lignes de A

— Fusion de matrices $D = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$.

Matrice $h^{S,t+1}$. Rappelons que $h^{S,t+1} = \begin{pmatrix} -h^S & h^S i^B h^{S,t} \\ 0 & h^{S,t} \end{pmatrix}$. En particulier, cette matrice est calculée comme une modification de $h^{S,t}$. Ses caractéristiques matricielles sont :

— $|B_{C^B}| + |B_{C^{B,t}}|$ lignes et $|B_{C^B}| + |B_{C^{B,t}}|$ colonnes. Or, $|B_{C^B}|$ est "petit" par rapport à $|B_{C^{B,t}}|$, d'après l'hypothèse (ii) du paragraphe 2.1.1-a. Dit autrement, le nombre de lignes et de colonnes ajoutées à l'issue de la modification de $h^{S,t}$ est "petit" par rapport à $|B_{C^{B,t}}|$.

— on l'étudie comme une variation de la matrice nulle ;

— il n'est pas possible d'exprimer $h_{\Delta}^{S,t+1}$ dans le cas général car $h^S i^B h^{S,t}$ n'est pas connue.

Observons que :

— h^S **est multipliée par** -1 ;

— pour le produit $h^S i^B h^{S,t}$,

· l'ordre n'importe pas car il existe dans tous les cas une dimension exploitable (cf paragraphe 2.1.1-d) ;

▷ dans le cas $(h^S i^B)h^{S,t}$, les $|B_{C^B}|$ lignes de h^S sont exploitables pour calculer $(h^S i^B)$ et $(h^S i^B)h^{S,t}$;

▷ dans le cas $h^S(i^B h^{S,t})$, les $|B_{C^B}|$ lignes de i^B sont exploitables pour calculer $(i^B h^{S,t})$ et les $|B_{C^B}|$ lignes de $h^{S,t}$ sont exploitables pour calculer $h^S(i^B h^{S,t})$;

· **le produit** $i^B h^{S,t}$ **doit se faire sans parcourir :**

▷ les $|B_{C^{B,t}}|$ colonnes de i^B ;

▷ les $|B_{C^{B,t}}|$ lignes et $|B_{C^{S,t}}|$ colonnes de $h^{S,t}$

— calculer $h^{S,t+1}$ revient à fusionner 4 matrices (cf. paragraphe 2.1.1-e),

· par **modification de la matrice en bas à droite**, à savoir $h^{S,t}$;

· sans parcourir les $|B_{C^{B,t}}|$ colonnes de $h^S i^B h^{S,t}$ pour éviter une dépendance à $|B_{C^{B,t}}| \geq |B_{C^t}|$, et donc au nombre de cellules de la structure topologique.

Prérequis d'implémentation.

— Représentation en matrice creuse : les valeurs nulles sont implicites ;

— Produit d'une matrice avec un scalaire ;

— Produit matriciel $A \times B$ en parcourant les lignes de A

— Fusion de matrices $D = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$.

2.1.3-c Récapitulatif des prérequis d'implémentation pour l'identification.

On constate que deux calculs sont récurrents dans l'application du théorème SECE à l'identification : le **produit d'une matrice** avec une matrice ou un scalaire, et la **fusion de matrices**. De plus, nous avons vu qu'il doit être possible de **construire des nouvelles matrices** et de **modifier des matrices existantes**, par exemple en supprimant une colonne. Enfin, la totalité des **matrices sont creuses** et, pour certaines, **l'identité est implicite**. Ce paragraphe reprend l'ensemble des prérequis d'implémentation constatés dans le cas de l'identification. Pour chaque prérequis, nous rappelons les matrices dont il émane.

Remarque. Sur support numérique, les noms de matrice de ce paragraphe sont des liens hypertextes. Ils redirigent vers l'analyse les concernant si l'on clique dessus.

Structure de donnée.

Structure-A | Représentation en matrice creuse : les valeurs nulles sont implicites. C'est le cas de la totalité des matrices manipulées.

Structure-B | Représentation implicite de l'identité : $\phi[x, x] = 1$ est implicite. C'est le cas de : g^{t+1}, f^{t+1}, $g^{S,t+1}$, $f^{S,t+1}$.

Construction de matrices.

Construction-B | Construction d'une matrice qui ne dépend pas de son nombre de colonnes. Cette construction est nécessaire pour le calcul de : i^B, i^S, $f^{S,t+1}$.

Construction-D | Construction d'une matrice qui ne dépend pas de son nombre de lignes. Cette construction est nécessaire pour le calcul de : g^{t+1} ;

Modification de matrices.

Modification-A | Suppression d'une ligne d'une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes. Cette modification est nécessaire pour le calcul de : g^{t+1}, ∂^{t+1}.

Modification-B | Somme de deux colonnes dans une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes. Cette modification est nécessaire pour le calcul de : ∂^{t+1}, f^{t+1}.

Modification-C | Suppression d'une colonne d'une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes. Cette modification est nécessaire pour le calcul de : ∂^{t+1}, f^{t+1}.

Modification-D | Ajout de lignes vides dans une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes. Cette modification est nécessaire pour le calcul de : f^{t+1}.

Produit impliquant une matrice.

Produit-A | Produit matriciel $A \times B$ en parcourant les lignes de A. Ce produit est nécessaire pour le calcul de : i^B, i^S, $f^{S,t+1}$, $g^{S,t+1}$, $h^{S,t+1}$.

Produit-B | Produit matriciel $A \times B$ en parcourant les colonnes de B. Ce produit est nécessaire pour le calcul de : g^{t+1}, h^{t+1}.

Produit-C | Produit d'une matrice avec un scalaire. Ce produit est nécessaire pour le calcul de : $\partial^{B,t+1}$, $\partial^{S,t+1}$, $h^{S,t+1}$.

Produit-D | Produit d'une matrice avec un scalaire, sans dépendre de son nombre de lignes. Ce produit est nécessaire pour le calcul de : g^{t+1} ;

Produit-E | Produit d'une matrice avec un scalaire, sans dépendre de son nombre de colonnes. Ce produit est nécessaire pour le calcul de : $g^{S,t+1}$;

Fusion de matrices.

Fusion-A | Fusion de matrices $D = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$. Cette fusion est nécessaire pour le calcul de : $\partial^{B,t+1}$, $\partial^{S,t+1}$, $f^{S,t+1}$,$g^{S,t+1}$, h^{t+1}, $h^{S,t+1}$.

Fusion-B | Fusion de matrices $B = \begin{bmatrix} A & B \end{bmatrix}$. Cette fusion est nécessaire pour le calcul de : g^{t+1}.

2.1.4 Désidentification

Dans ce paragraphe, nous considérons une SECE

$$(C, \partial) \underset{\mathsf{r}}{\overset{\mathsf{i}}{\longleftrightarrow}} (C^{t+1}, \partial^{t+1}) \underset{\mathsf{s}}{\overset{\mathsf{j}}{\longleftrightarrow}} (C^t, \partial^t)$$

induite par une désidentification, et l'équivalence homologique

$$\gamma^t : (C^t, \partial^t) \overset{\rho^t}{\Longleftarrow} (C^{B,t}, \partial^{B,t}) \overset{\rho^{S,t}}{\Longrightarrow} (C^{S,t}, \partial^{S,t})$$

Nous analysons le calcul de l'équivalence homologique γ^{t+1} par application du théorème SECE, et par modification de γ^t sans émettre d'hypothèses sur les matrices ∂^t, $\partial^{B,t}$, $\partial^{S,t}$ ou sur les réductions ρ, ρ^S, ρ^t et $\rho^{S,t}$. L'objectif de ce paragraphe est d'établir l'ensemble des prérequis d'implémentation qui doivent être satisfaits pour optimiser le calcul de l'équivalence homologique γ^{t+1} lorsque le passage de l'étape t à l'étape $t+1$ se fait par désidentification. Optimiser ce calcul revient à profiter du "petit" nombre de cellules impactées par la désidentification (cf hypothèse (ii) du paragraphe 2.1.1-a). Nous distinguons deux cas :

— le cas où les opérations du processus de construction étudié ne sont que des désidentifications. Dans ce cas, seules quelques matrices sont calculées ;

— le cas où les opérations du processus de construction sont des désidentifications ou des identifications. Dans ce cas, toutes les matrices sont calculées.

Remarque. Rappelons que nous avons fait le choix de caractériser une désidentification par l'identification inverse qu'elle induit (cf 1.2.1-b et 1.2.2-a). La SECE induite par une désidentification est étudiée à partir de cette identification inverse et les cellules de $B_{C^{t+1}}$ sont dites survivantes ou non-survivantes en fonction de cette identification. Enfin, notons que contrairement à l'identification, le fait de calculer ou non les générateurs d'homologie n'induit pas de cas particuliers pour la désidentification.

2.1.4-a Processus de construction composé uniquement de désidentifications

Les matrices suivantes sont calculées : $\partial^{t+1}, \chi, \chi^B, \chi^S, \partial^{B,t+1}, \partial^{S,t+1}, f^{t+1}, g^{S,t+1}$. La figure 2.8 reprend les différents éléments manipulés dans le calcul de γ^{t+1}. Les caractéristiques des matrices connues au début du calcul de γ^{t+1} sont celles répertoriées dans le tableau 2.1. La figure 2.9 reprend les différentes formules du calcul de γ^{t+1} et illustre les dépendances qu'il existe entre les matrices manipulées. Analysons ces matrices.

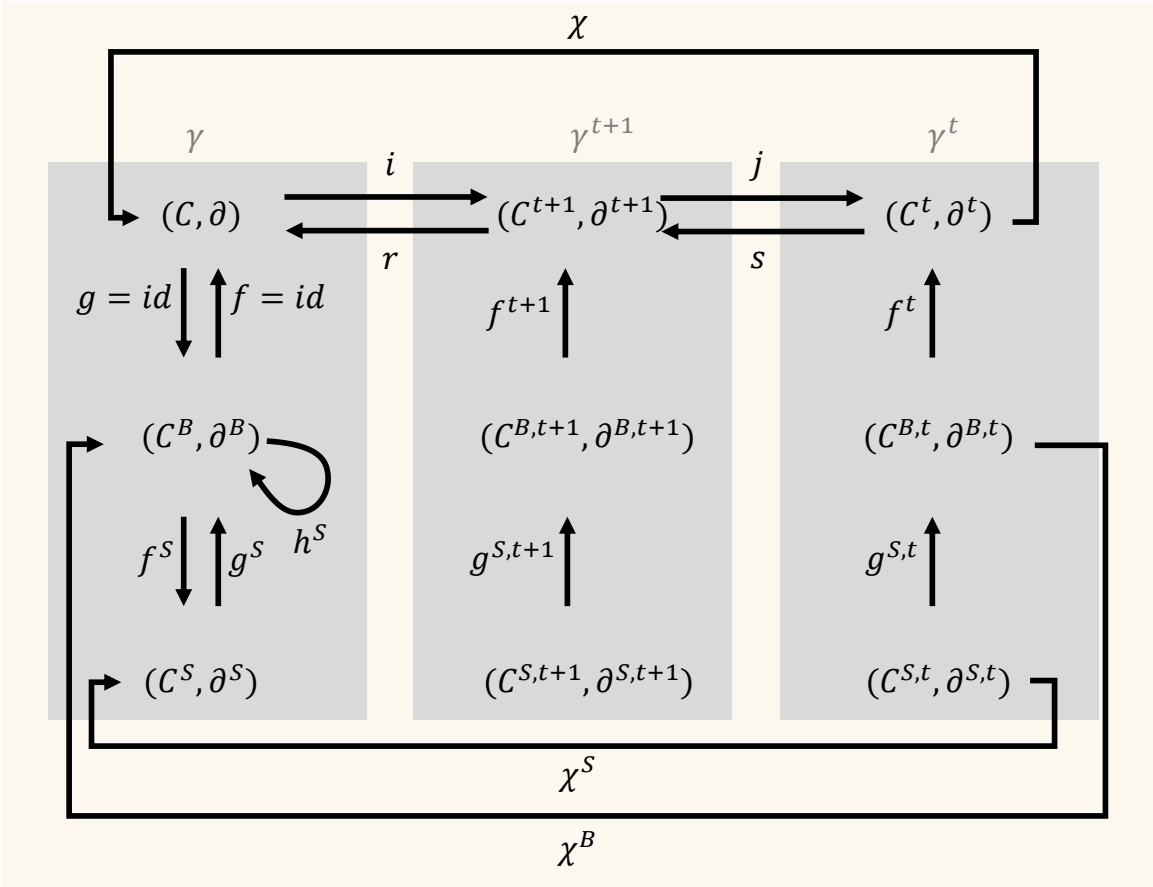


FIGURE 2.8 – Représentation des éléments manipulés dans l'application du théorème SECE à la désidentification, que les générateurs d'homologie puissent être calculés ou non. Les équivalences homologiques sont mises en évidence via un fond gris.

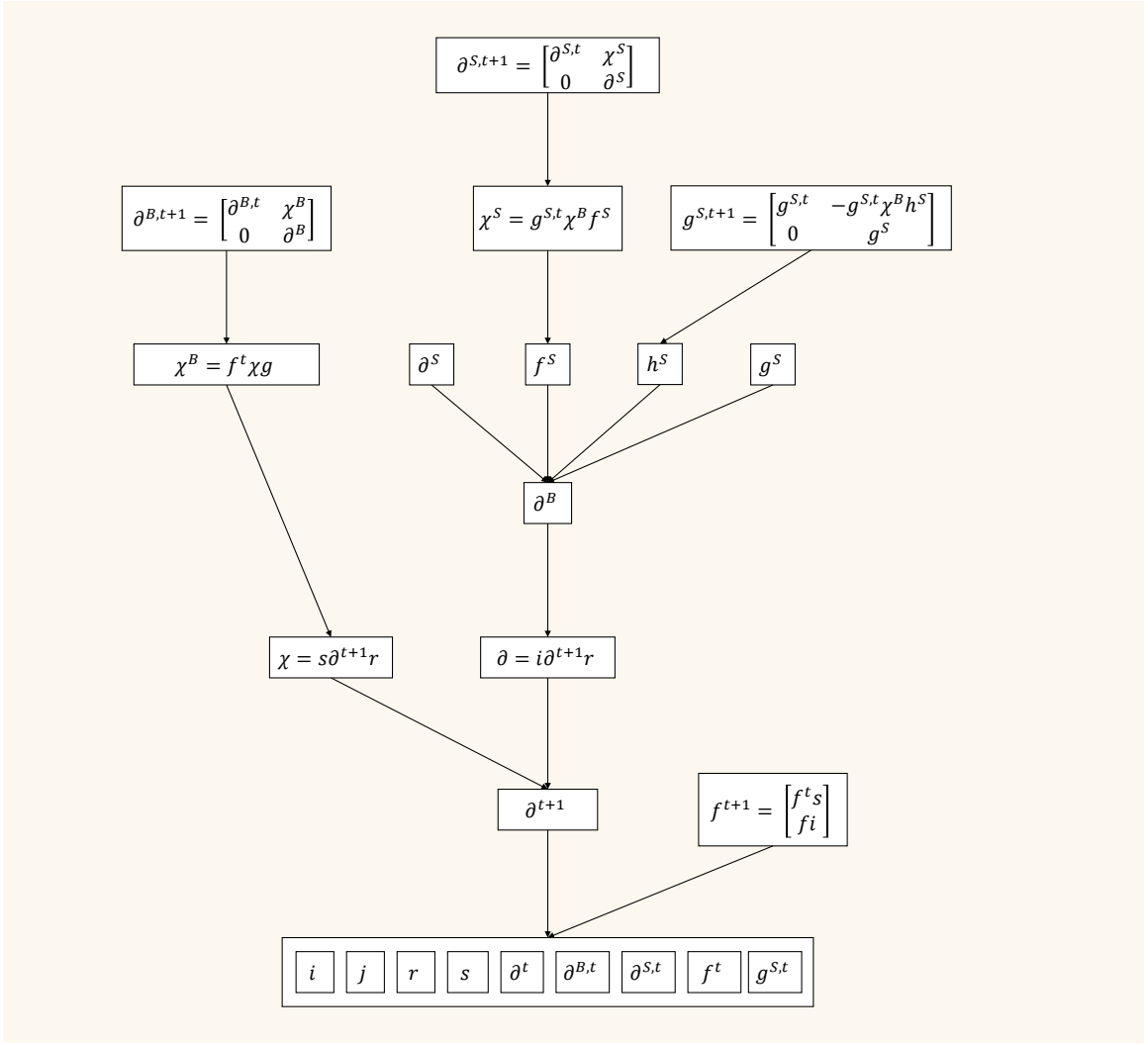


FIGURE 2.9 – Graphe de dépendance des matrices utilisées lors de l'application du théorème SECE à la désidentification, que les générateurs d'homologie puissent être calculés ou non. Une flèche d'un nœud A vers un nœud B indique que A a besoin de B pour son calcul. Pour simplifier le schéma, toutes les relations ne sont pas représentées. Les matrices $f = id$, $g = id$ et $h = 0$ ne sont pas représentées.

Matrice ∂^{t+1}. Cette matrice représente le bord du complexe de chaînes associé à la structure topologique de l'étape $t+1$, c'est-à-dire après que la désidentification lui ait été appliquée. Elle est calculée comme une modification de ∂^t. Pour chaque p-cellule σ' créée dans $(C^{t+1}, \partial^{t+1})$ par éclatement d'une p-cellule σ dans (C^t, ∂^t) :

— on **duplique** la ligne et la colonne σ en σ et σ'. Cette opération représente l'ajout de la cellule σ' dans $B_{C^{t+1}}$;

— on **supprime** certains coefficients de la matrice obtenue après duplication. Cette opération représente la modification de l'étoile des cellules éclatées.

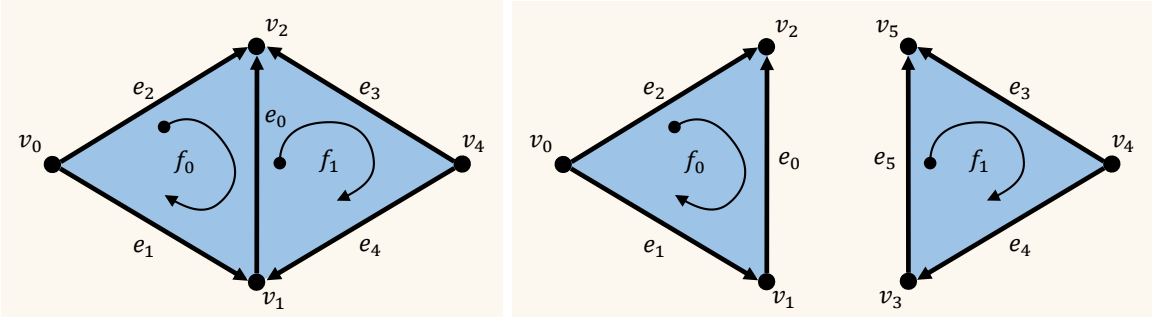
Remarque. Les coefficients supprimés dépendent de la modification engendrée par la désidentification sur le bord des cellules. Ils se déduisent des informations supplémentaires qu'il faut fournir, en plus de I et ζ, pour caractériser une désidentification.

Prérequis d'implémentation.

— Duplication d'une ligne d'une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes ;

— Duplication d'une colonne d'une matrice sans dépendre de son nombre de lignes ni de son nombre de colonnes ;

— Suppression de coefficients dans une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes ;



(a) (b)

FIGURE 2.10 – Représentation graphique des complexes de chaînes (C^t, ∂^t) et $(C^{t+1}, \partial^{t+1})$ associés à un ensemble semi-simplicial sur lequel est appliquée une désidentification. Les sommets v_2 et v_1 sont éclatés, ainsi que l'arête e_0. (a) (C^t, ∂^t). (b) $(C^{t+1}, \partial^{t+1})$.

Exemple. Cet exemple illustre les modifications induites par une désidentification au niveau du complexe de chaînes (C^t, ∂^t) de la figure 2.10, associé à un ensemble semi-simplicial. La désidentification est caractérisée par $I = (I^p)^{0 \leq p \leq 2}$:

$$I^0 = \{\{v_0\}, \{v_1, v_3\}, \{v_2, v_5\}, \{v_4\}\}$$
$$I^1 = \{\{e_0, e_5\}, \{e_1\}, \{e_2\}, \{e_3\}, \{e_4\}\}$$
$$I^2 = \{\{f_0\}, \{f_1\}\}$$

et $v_0\zeta = v_0$, $v_1\zeta = v_3\zeta = v_1$, $v_2\zeta = v_5\zeta = v_2$, $e_0\zeta = e_5\zeta = e_0$. La [8] matrice de bord ∂^t est :

∂^t	v_0	v_1	v_2	v_4	e_0	e_1	e_2	e_3	e_4	f_0	f_1
v_0	.	.	.	.	.	.	.	.	.	.	.
v_1	.	.	.	.	.	.	.	.	.	.	.
v_2	.	.	.	.	.	.	.	.	.	.	.
v_4	.	.	.	.	.	.	.	.	.	.	.
e_0	.	-1	1	.	.	.	.	.	.	.	.
e_1	-1	1	.	.	.	.	.	.	.	.	.
e_2	-1	.	1	.	.	.	.	.	.	.	.
e_3	.	.	1	-1	.	.	.	.	.	.	.
e_4	.	1	.	-1	.	.	.	.	.	.	.
f_0	.	.	.	.	-1	-1	1	.	.	.	.
f_1	.	.	.	.	1	.	.	-1	1	.	.

Modifions ∂^t en dupliquant les lignes et colonnes :

— v_1 en v_1 et v_3 ;

— v_2 en v_2 et v_5 ;

— e_0 en e_0 et e_5.

La matrice ainsi obtenue est :

	v_0	v_1	v_2	$v_3(v_1)$	v_4	$v_5(v_2)$	e_0	e_1	e_2	e_3	e_4	$e_5(e_0)$	f_0	f_1
v_0	.	.	.	.	.	.	.	.	.	.	.	.	.	.
v_1	.	.	.	.	.	.	.	.	.	.	.	.	.	.
v_2	.	.	.	.	.	.	.	.	.	.	.	.	.	.
$v_3(v_1)$	.	.	.	.	.	.	.	.	.	.	.	.	.	.
v_4	.	.	.	.	.	.	.	.	.	.	.	.	.	.
$v_5(v_2)$	.	.	.	.	.	.	.	.	.	.	.	.	.	.
e_0	.	-1	1	-1	.	1	.	.	.	.	.	.	.	.
e_1	-1	1	.	1	.	.	.	.	.	.	.	.	.	.
e_2	-1	.	1	.	.	1	.	.	.	.	.	.	.	.
e_3	.	.	1	.	-1	1	.	.	.	.	.	.	.	.
e_4	.	1	.	1	-1	.	.	.	.	.	.	.	.	.
$e_5(e_0)$	.	-1	1	-1	.	1	.	.	.	.	.	.	.	.
f_0	.	.	.	.	.	.	-1	-1	1	.	.	-1	.	.
f_1	.	.	.	.	.	.	-1	.	.	-1	1	1	.	.

En supprimant les coefficients $[e_0, v_5]$, $[e_0, v_3]$, $[e_1, v_3]$, $[e_2, v_5]$, $[e_3, v_2]$, $[e_4, v_1]$, $[e_5, v_1]$, $[e_5, v_2]$,

8. Pour simplifier l'exemple, on regroupe les différentes matrices de bord $(\partial^{t,p})^{0\leq p\leq 2}$ en une seule.

$[f_0, e_5]$ et $[f_1, e_0]$ de cette matrice, on obtient ∂^{t+1} :

∂^{t+1}	v_0	v_1	v_2	$v_3(v_1)$	v_4	$v_5(v_2)$	e_0	e_1	e_2	e_3	e_4	$e_5(e_0)$	f_0	f_1
v_0	.	.	.	.	.	.	.	.	.	.	.	.	.	.
v_1	.	.	.	.	.	.	.	.	.	.	.	.	.	.
v_2	.	.	.	.	.	.	.	.	.	.	.	.	.	.
$v_3(v_1)$	.	.	.	.	.	.	.	.	.	.	.	.	.	.
v_4	.	.	.	.	.	.	.	.	.	.	.	.	.	.
$v_5(v_2)$	.	.	.	.	.	.	.	.	.	.	.	.	.	.
e_0	.	-1	1	.	.	.	.	.	.	.	.	.	.	.
e_1	-1	1	.	.	.	.	.	.	.	.	.	.	.	.
e_2	-1	.	1	.	.	.	.	.	.	.	.	.	.	.
e_3	.	.	.	.	-1	1	.	.	.	.	.	.	.	.
e_4	.	.	.	1	-1	.	.	.	.	.	.	.	.	.
$e_5(e_0)$	.	.	.	-1	.	1	.	.	.	.	.	.	.	.
f_0	.	.	.	.	.	.	-1	-1	1	.	.	.	.	.
f_1	.	.	.	.	.	.	.	.	.	-1	1	1	.	.

Matrice χ. Rappelons que $\chi = s\partial^{t+1}r$. Ses caractéristiques matricielles sont :

— $|B_{C^t}|$ lignes et $|B_C|$ colonnes. Notons que $|B_C|$ est "petit" par rapport à $|B_{C^t}|$, d'après l'hypothèse (ii) du paragraphe 2.1.1-a ;

— on l'étudie comme une variation de la matrice nulle ;

— il n'est pas possible d'étudier précisément χ_Δ dans le cas général car nous n'avons pas d'hypothèse sur ∂^{t+1}.

Observons que :

— la matrice χ est construite à chaque étape de construction, et **cette construction ne doit pas dépendre de ses $|B_{C^t}|$ lignes** ;

— ∂^{t+1} est nécessaire au calcul. Dit autrement, pour calculer χ, il faut modifier ∂^t en fonction de la désidentification appliquée à la structure topologique, et donc s'assurer qu'aucune autre matrice n'ait besoin de ∂^t dans son calcul (ce qui est le cas).

— **l'ordre du produit importe**, il doit être $s(\partial^{t+1}r)$ car les $|B_C|$ colonnes de r représentent la seule dimension exploitable des matrices du produit (cf. paragraphe 2.1.1-d) ;

— le produit avec s revient à supprimer les lignes de $(\partial^{t+1}r)$ qui représentent une cellule non survivante de $B_{C^{t+1}}$. Il n'est donc pas nécessaire de représenter explicitement la matrice s ici.

Prérequis d'implémentation.

— Représentation en matrice creuse : les valeurs nulles sont implicites ;

— Construction d'une matrice qui ne dépend pas de son nombre de lignes ;

— Produit matriciel $A \times B$ en parcourant les colonnes de B ;

Matrice χ^B. Rappelons que $\chi^B = f^t \chi g$ et que g est toujours l'identité (cf. paragraphe 2.1.2-b). On peut donc simplifier ce calcul en $\chi^B = f^t \chi$. Ses caractéristiques matricielles sont :

- $|B_{C^{B,t}}|$ lignes et $|B_{C^B}|$ colonnes. Notons que $|B_{C^B}|$ est "petit" par rapport à $|B_{C^{B,t}}|$, d'après l'hypothèse (ii) du paragraphe 2.1.1-a
- on l'étudie comme une variation de la matrice nulle ;
- il n'est pas possible d'étudier précisément χ^B_Δ dans le cas général car nous n'avons pas d'hypothèse sur f^t.

Observons que :

- la matrice χ^B est construite à chaque étape de construction, et **cette construction ne doit pas dépendre de ses $|B_{C^{B,t}}|$ lignes** ;
- le produit $f^t \chi$ **se fait en parcourant les** $|B_{C^B}|$ colonnes de χ ;

Prérequis d'implémentation.

- Représentation en matrice creuse : les valeurs nulles sont implicites ;
- Construction d'une matrice qui ne dépend pas de son nombre de lignes ;
- Produit matriciel $A \times B$ en parcourant les colonnes de B ;

Matrice χ^S. Rappelons que $\chi^S = g^{S,t} \chi^B f^S$. Ses caractéristiques matricielles sont :

- $|B_{C^{S,t}}|$ lignes et $|B_{C^S}|$ colonnes. Notons que $|B_{C^S}|$ est "petit" par rapport à $|B_{C^{S,t}}|$, d'après l'hypothèse (ii) du paragraphe 2.1.1-a ;
- on l'étudie comme une variation de la matrice nulle ;
- il n'est pas possible d'étudier précisément χ^S_Δ dans le cas général car nous n'avons pas d'hypothèse sur $g^{S,t}$.

Observons que :

- la matrice χ^S est construite à chaque étape de construction, et **cette construction ne doit pas dépendre de ses $|B_{C^{S,t}}|$ lignes** ;
- **l'ordre du produit importe**, il doit être $g^{S,t}(\chi^B f^S)$ car les dimensions exploitables sont celles de f^S, à savoir $|B_{C^B}|$ lignes et $|B_{C^S}|$ colonnes ;

Prérequis d'implémentation.

- Représentation en matrice creuse : les valeurs nulles sont implicites ;
- Construction d'une matrice qui ne dépend pas de son nombre de lignes ;
- Produit matriciel $A \times B$ en parcourant les colonnes de B.

Matrice $\partial^{B,t+1}$. Rappelons que $\partial^{B,t+1} = \begin{pmatrix} \partial^{B,t} & \chi^B \\ 0 & \partial^B \end{pmatrix}$. En particulier, cette matrice est une modification de la matrice $\partial^{B,t}$. Ses caractéristiques matricielles sont :

- $|B_{C^{B,t}}|+|B_{C^B}|$ lignes et $|B_{C^{B,t}}|+|B_{C^B}|$ colonnes. Notons que $|B_{C^B}|$ est "petit" par rapport à $|B_{C^{B,t}}|$, d'après l'hypothèse (ii) du paragraphe 2.1.1-a. Dit autrement, le nombre de lignes et de colonnes ajoutées à l'issue de la modification de $\partial^{B,t}$ est "petit" ;
- on l'étudie comme une variation de la matrice nulle ;

— $\partial_\Delta^{B,t+1} = \partial_\Delta^{B,t} + \chi_\Delta^B + \partial_\Delta^B$. Dit autrement, le calcul de $\partial^{B,t+1}$ par modification de $\partial^{B,t}$ implique l'ajout de χ_Δ^B et ∂_Δ^B valeurs explicites dans $\partial^{B,t+1}$.

Observons que :

— calculer $\partial^{B,t+1}$ revient à **fusionner** 4 **matrices** (cf. paragraphe 2.1.1-e), dont l'une est une matrice nulle. En particulier, cette fusion se fait :

· **par modification de la matrice en haut à gauche**, à savoir $\partial^{B,t}$;

· **sans parcourir les** $|B_{C^{B,t}}|$ **lignes de** χ^B, pour éviter une dépendance à $|B_{C^{B,t}}| \geq |B_{C^t}|$, et donc au nombre de cellules de la structure topologique.

Prérequis d'implémentation.

— Représentation en matrice creuse : les valeurs nulles sont implicites ;

— Fusion de matrices $A = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$.

Matrice $\partial^{S,t+1}$**.** Rappelons que $\partial^{S,t+1} = \begin{pmatrix} \partial^{S,t} & \chi^S \\ 0 & \partial^S \end{pmatrix}$. En particulier, cette matrice est une modification de $\partial^{S,t}$. Ses caractéristiques matricielles sont :

— $|B_{C^{S,t}}|+|B_{C^S}|$ lignes et $|B_{C^{S,t}}|+|B_{C^S}|$ colonnes. Notons que $|B_{C^S}|$"petit" d'après l'hypothèse (ii) du paragraphe 2.1.1-a. Dit autrement, le nombre de lignes et de colonnes ajoutées à l'issue de la modification de $\partial^{S,t}$ est "petit" ;

— on l'étudie comme une variation de la matrice nulle ;

— $\partial_\Delta^{S,t+1} = \partial_\Delta^{S,t} + \chi_\Delta^S + \partial_\Delta^S$. Dit autrement, le calcul de $\partial^{S,t+1}$ par modification de $\partial^{S,t}$ implique l'ajout de χ_Δ^S et ∂_Δ^S valeurs explicites dans $\partial^{S,t+1}$;

Observons que :

— calculer $\partial^{S,t+1}$ revient à **fusionner** 4 **matrices** (cf. paragraphe 2.1.1-e), dont l'une est une matrice nulle. En particulier, cette fusion se fait :

· **par modification de la matrice en haut à gauche**, à savoir $\partial^{S,t}$;

· **sans parcourir les** $|B_{C^{S,t}}|$ **lignes de** χ^S, pour éviter une dépendance à $|B_{C^{S,t}}| \leq |B_{C^{B,t}}|$, et donc au nombre de cellules de la structure topologique.

Prérequis d'implémentation.

— Représentation en matrice creuse : les valeurs nulles sont implicites ;

— Fusion de matrices $A = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$.

Matrice f^{t+1}**.** Rappelons que $f^{t+1} = \begin{pmatrix} f^t s \\ fi \end{pmatrix}$ et que f est toujours l'identité (cf. paragraphe 2.1.2-b). On peut donc simplifier ce calcul en $f^{t+1} = \begin{pmatrix} f^t s \\ i \end{pmatrix}$. En particulier, cette matrice est une modification de f^t. Ses caractéristiques matricielles sont :

— $|B_{C^t}| + |B_C|$ lignes et $|B_{C^t}|$ colonnes. Notons que $|B_C|$ est "petit" par rapport à $|B_{C^t}|$, d'après l'hypothèse (ii) du paragraphe 2.1.1-a. Dit autrement, le nombre de lignes ajoutées à l'issue de la modification de f^t est "petit" ;

— on l'étudie comme une variation de l'identité ;

— il n'est pas possible d'étudier précisément f^{t+1} dans le cas général car nous n'avons pas d'hypothèse sur f^t.

Observons que :

— pour le produit $f^t s$,

· aucune des dimensions des deux matrices n'est exploitable. Ce calcul ne peut pas être fait sous la forme d'un produit matriciel ;

· s représente l'inclusion de C^t dans C^{t+1}. Dit autrement, seules les cellules de $B_{C^{t+1}}$ ont un antécédent par s dans B_{C^t} ;

· ce calcul revient à **ajouter des colonnes vides dans** f^t, pour chaque cellule non survivante de $B_{C^{t+1}}$;

— calculer f^{t+1} implique de fusionner 2 matrices, à savoir $f^t s$ et i. Pour ce faire, on commence par calculer $f^t j$ (par modification de f^t), puis on fusionne i en parcourant ses $|B_C|$ lignes.

Prérequis d'implémentation.

— Représentation en matrice creuse : les valeurs nulles sont implicites ;

— Représentation implicite de l'identité : $\phi[x, x] = 1$ est implicite ;

— Ajout de colonnes vides dans une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes ;

— Fusion de matrices $A = \begin{bmatrix} A \\ B \end{bmatrix}$.

Matrice $g^{S,t+1}$. Rappelons que $g^{S,t+1} = \begin{pmatrix} g^{S,t} & -g^{S,t}\chi^B h^S \\ 0 & g^S \end{pmatrix}$. En particulier, $g^{S,t+1}$ est une modification de $g^{S,t}$. Ses caractéristiques matricielles sont :

— $|B_{C^{S,t}}| + |B_{C^S}|$ lignes et $|B_{C^{S,t}}| + |B_{C^S}|$ colonnes. Notons que $|B_{C^S}|$ est "petit" par rapport à $|B_{C^{S,t}}|$, d'après l'hypothèse (ii) du paragraphe 2.1.1-a. Dit autrement, le nombre de lignes et de colonnes ajoutées à l'issue de la modification de $g^{S,t}$ est "petit" ;

— on l'étudie comme une variation de l'identité ;

— il n'est pas possible d'étudier précisément $g^{S,t+1}$ dans le cas général car nous n'avons pas d'hypothèse sur $g^{S,t}$.

Observons que :

— pour le produit $-g^{S,t}\chi^B h^S$,

· la matrice qui résulte de ce produit doit être **construite sans dépendre de ses** $|B_{C^{S,t}}|$ **lignes** ;

· la matrice qui résulte de ce produit **est multipliée par** -1 ;

· **l'ordre du produit importe** et doit être $-g^{S,t}(\chi^B h^S)$ car aucune des dimensions de $g^{S,t}$ n'est exploitable pour le produit (cf. paragraphe 2.1.1-d). Plus précisément, ce produit se fait en parcourant les $|B_{C^B}|$ colonnes de h^S ;

— il faut fusionner 4 matrices dont l'une est nulle (cf. paragraphe 2.1.1-e). En particulier, cette fusion se fait **par modification de la matrice en haut à gauche**, à savoir $g^{S,t}$.

Prérequis d'implémentation.

— Représentation en matrice creuse : les valeurs nulles sont implicites ;

— Représentation implicite de l'identité : $\phi[x, x] = 1$ est implicite ;

— Construction d'une matrice qui ne dépend pas de son nombre de lignes ;

— Produit matriciel $A \times B$ en parcourant les colonnes de B ;

— Produit d'une matrice avec un scalaire ;

— Fusion de matrices $A = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$.

2.1.4-b Processus de construction plus général.

Lorsque le processus de construction étudié est plus général, c'est-à-dire qu'il composé d'identifications et de désidentifications, toutes les matrices doivent être calculées. En particulier, les matrices g^t et $f^{S,t}$ doivent être calculées car elles sont nécessaires aux étapes de construction où l'opération est une identification (que les générateurs d'homologie puissent être calculés ou non). Les matrices supplémentaires sont : $g^{t+1}, h^{t+1}, f^{S,t+1}, h^{S,t+1}$. La figure 2.11 reprend les différents éléments manipulés dans le calcul de γ^{t+1}. Les caractéristiques des matrices connues au début de ce calcul sont répertoriées dans 2.2. La figure 2.12 reprend les différentes formules du calcul de γ^{t+1} et illustre les dépendances qu'il existe entre les matrices manipulées. Analysons ces matrices.

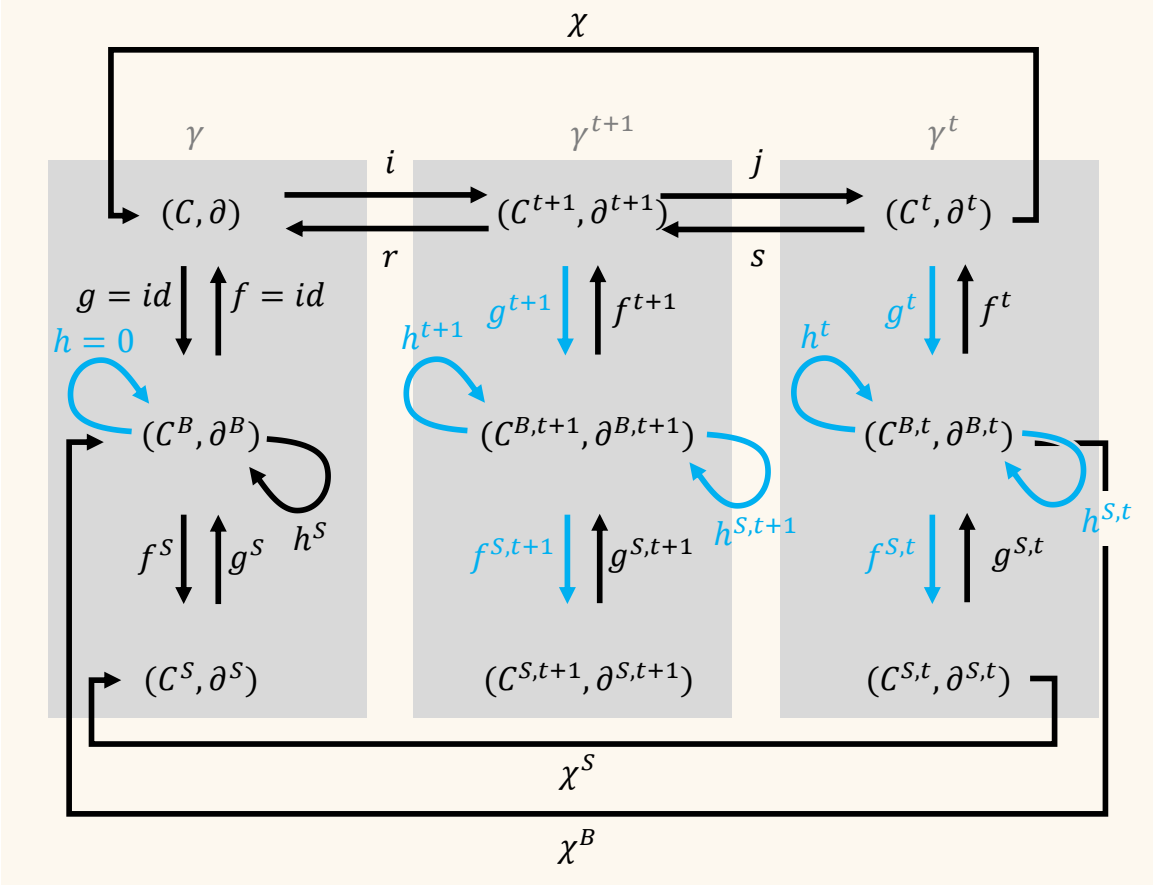


FIGURE 2.11 – Représentation des éléments impliqués dans le théorème SECE dans le cas de la désidentification lorsque les générateurs d'homologie sont calculés. Les équivalences homologiques sont mises en évidence via un fond gris. En bleu, les éléments supplémentaires calculés lorsque les générateurs d'homologie le sont.

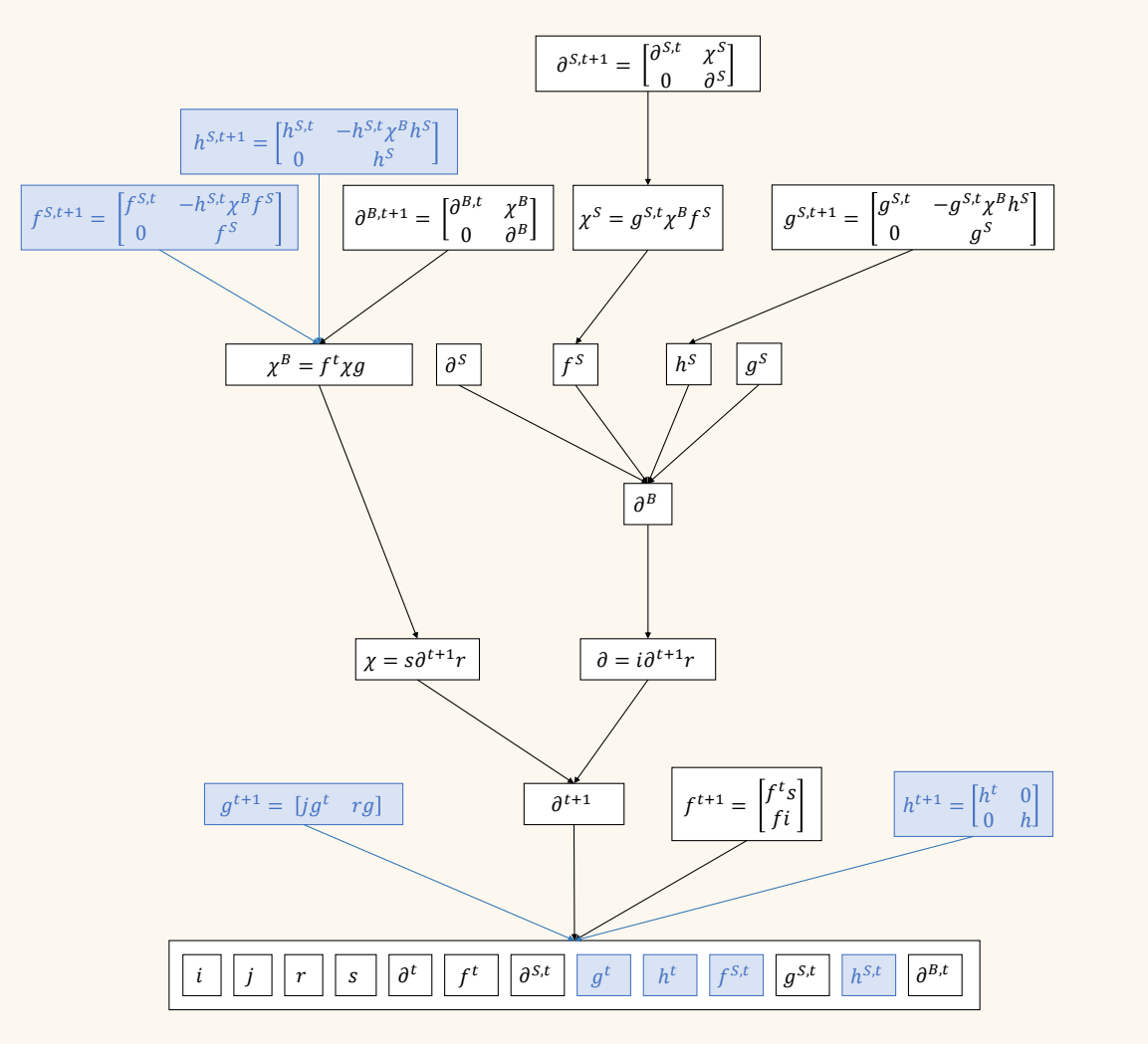


FIGURE 2.12 – Graphe de dépendance des matrices utilisées dans l'application du théorème SECE à la désidentification, lorsque les générateurs d'homologie sont calculés. Toutes les relations ne sont pas représentées. Les matrices $f = id$, $g = id$ et $h = 0$ ne sont pas représentées. En bleu, les matrices supplémentaires par rapport au cas sans générateurs.

Matrice g^{t+1}. Rappelons que $g^{t+1} = \begin{pmatrix} jg^t & rg \end{pmatrix}$ et que g est toujours l'identité (cf. paragraphe 2.1.2-b). On peut donc simplifier ce calcul en $g^{t+1} = \begin{pmatrix} jg^t & r \end{pmatrix}$. En particulier g^{t+1} est une modification de la matrice g^t. Ses caractéristiques matricielles sont :

— $|B_{C^{t+1}}|$ lignes et $|B_{C^{B,t}}| + |B_{C^B}|$ colonnes. Notons que $|B_{C^B}|$ est "petit" par rapport à $|B_{C^{B,t}}|$, d'après l'hypothèse (ii) du paragraphe 2.1.1-a. Dit autrement, le nombre de colonnes ajoutées à l'issue de la modification de g^t est "petit" ;

— on l'étudie comme une variation de l'identité ;

— il n'est pas possible d'étudier précisément g^{t+1}_Δ dans le cas général car nous n'avons pas d'hypothèse sur g^t

Observons que :

— pour le produit jg^t,

  · aucune des dimensions des deux matrices n'est exploitable. Ce calcul ne peut pas être fait sous la forme d'un produit matriciel ;

  · j est l'identité pour toutes les cellules survivantes de $B_{C^{t+1}}$, et j associe à chaque cellule non-survivante de $B_{C^{t+1}}$ la cellule survivante avec laquelle elle est identifiée ;

  · le produit jg^t revient à **dupliquer** $|B_C|$ **lignes** dans g^t. En particulier, une ligne est dupliquée pour chaque cellule non-survivante, celle qui représente la cellule survivante avec laquelle elle est identifiée ;

— calculer g^{t+1} implique de fusionner 2 matrices (cf. paragraphe 2.1.1-e), à savoir jg^t et r. Pour ce faire, on commence par calculer jg^t (par modification de g^t), puis on fusionne r.

Prérequis d'implémentation.

— Représentation en matrice creuse : les valeurs nulles sont implicites ;

— Représentation implicite de l'identité : $\phi[x, x] = 1$ est implicite ;

— Duplication d'une ligne d'une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes ;

— Fusion de matrices $A = \begin{bmatrix} A & B \end{bmatrix}$.

Matrice h^{t+1}. Rappelons que $h^{t+1} = \begin{pmatrix} h^t & 0 \\ 0 & h \end{pmatrix}$ et que h est toujours la matrice nulle (cf. paragraphe 2.1.2-b). On peut donc simplifier ce calcul en $h^{t+1} = \begin{pmatrix} h^t & 0 \\ 0 & 0 \end{pmatrix}$. En particulier, cette matrice est une modification de h^t. Ses caractéristiques matricielles sont :

— $|B_{C^B}| + |B_{C^{B,t}}|$ lignes et $|B_{C^B}| + |B_{C^{B,t}}|$ colonnes. Notons que $|B_{C^B}|$ est "petit" par rapport à $|B_{C^{B,t}}|$, d'après l'hypothèse (ii) du paragraphe 2.1.1-a. Dit autrement, le nombre de lignes et colonnes ajoutées à l'issue de la modification de h^t est "petit" ;

— on l'étudie comme une variation de la matrice nulle ;

— h^t_Δ est constant. Aucune valeur explicite n'est ajoutée pour calculer h^{t+1}.

Observons que le calcul de h^{t+1} se résume à ajouter $|B_{C^B}|$ lignes vides et $|B_{C^B}|$ colonnes vides dans h^t.

Prérequis d'implémentation.

— Représentation en matrice creuse : les valeurs nulles sont implicites ;
— Ajout de lignes vides dans une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes ;
— Ajout de colonnes vides dans une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes.

Matrice $f^{S,t+1}$. Rappelons que $f^{S,t+1} = \begin{pmatrix} f^{S,t} & -h^{S,t}\chi^B f^S \\ 0 & f^S \end{pmatrix}$. En particulier, $f^{S,t+1}$ est une modification de la matrice $f^{S,t}$. Ses caractéristiques matricielles sont :

— $|B_{C^t}|+|B_{C^S}|$ lignes et $|B_{C^{S,t}}|+|B_{C^S}|$ colonnes. Notons que $|B_{C^S}|$ est "petit" par rapport à $|B_{C^{S,t}}|$, d'après l'hypothèse (ii) du paragraphe 2.1.1-a. Dit autrement, le nombre de lignes et colonnes ajoutées à l'issue de la modification de $f^{S,t}$ est "petit" ;
— on l'étudie comme une variation de l'identité ;
— il n'est pas possible d'étudier précisément $f^{S,t+1}$ dans le cas général car nous n'avons pas d'hypothèse sur $h^{S,t}$.

Observons que :

— pour le produit $-h^{S,t}\chi^B f^S$,
  · la matrice qui résulte de ce produit doit être **construite sans dépendre de ses $|B_{C^{B,t}}|$ lignes** ;
  · la matrice qui résulte de ce produit **est multipliée par** -1 ;
  · **l'ordre du produit importe**, il doit être $-h^{S,t}(\chi^B f^S)$ car aucune des dimensions de $h^{S,t}$ n'est exploitable pour le produit (cf. paragraphe 2.1.1-d). Plus précisément, ce produit se fait en parcourant les $|B_{C^S}|$ colonnes de f^S ;
— il faut fusionner 4 matrices (cf. paragraphe 2.1.1-e), dont l'une est nulle. En particulier, cette fusion se fait **par modification de la matrice en haut à gauche**, à savoir $f^{S,t}$.

Prérequis d'implémentation.

— Représentation en matrice creuse : les valeurs nulles sont implicites ;
— Représentation implicite de l'identité : $\phi[x,x] = 1$ est implicite ;
— Construction d'une matrice qui ne dépend pas de son nombre de lignes ;
— Produit matriciel $A \times B$ en parcourant les colonnes de B ;
— Produit d'une matrice avec un scalaire ;
— Fusion de matrices $A = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$.

Matrice $h^{S,t+1}$. Rappelons que $h^{S,t+1} = \begin{pmatrix} h^{S,t} & -h^{S,t}\chi^B h^S \\ 0 & h^S \end{pmatrix}$. En particulier, $h^{S,t+1}$ est une modification de la matrice $h^{S,t}$. Ses caractéristiques matricielles sont :

— $|B_{C^{B,t}}|+|B_{C^B}|$lignes et $|B_{C^{B,t}}|+|B_{C^B}|$colonnes. Notons que $|B_{C^B}|$ est "petit" par rapport à $|B_{C^{B,t}}|$, d'après l'hypothèse (ii) du paragraphe 2.1.1-a. Dit autrement, le nombre de lignes et colonnes ajoutées à l'issue de la modification de $h^{S,t}$ est "petit" ;

— on l'étudie comme une variation de la matrice nulle ;

— il n'est pas possible d'étudier précisément $h^{S,t+1}$ dans le cas général car nous n'avons pas d'hypothèse sur $h^{S,t}$.

Observons que :

— pour le produit $-h^{S,t}\chi^B h^S$,

· la matrice qui résulte de ce produit doit être **construite sans dépendre de ses $|B_{C^{B,t}}|$ lignes** ;

· la matrice qui résulte de ce produit **est multipliée par -1** ;

· **l'ordre du produit importe**, il doit être $-h^{S,t}(\chi^B h^S)$ car aucune des dimensions de $h^{S,t}$ n'est exploitable pour le produit (cf. paragraphe 2.1.1-d). Plus précisément, ce produit se fait en parcourant les $|B_{C^B}|$ colonnes de h^S ;

— il faut fusionner 4 matrices dont l'une est nulle (cf. paragraphe 2.1.1-e). En particulier, cette fusion se fait **par modification de la matrice en haut à gauche**, à savoir $h^{S,t}$;

Prérequis d'implémentation.

— Représentation en matrice creuse : les valeurs nulles sont implicites ;

— Construction d'une matrice qui ne dépend pas de son nombre de lignes ;

— Produit matriciel $A \times B$ en parcourant les colonnes de B

— Produit d'une matrice avec un scalaire

— Fusion de matrices $A = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$.

2.1.4-c Récapitulatif des prérequis d'implémentation pour la désidentification.

On constate que les calculs récurrents dans l'application du théorème SECE à la désidentification sont les mêmes que pour son application à l'identification, à savoir : **le produit d'une matrice** avec une matrice ou un scalaire, et la **fusion de matrices**. Les prérequis concernant **la construction de nouvelles matrices** et la **modification de matrices existantes** sont également les mêmes : suppression de colonnes, ajout de colonnes etc... Enfin, comme pour l'identification, toutes les **matrices sont creuses** et, pour certaines, **l'identité est implicite**. Ce paragraphe reprend l'ensemble des prérequis d'implémentation constatés dans le cas de la désidentification. Pour chaque prérequis, nous rappelons les matrices dont il émane. Les prérequis propres à la désidentification sont en gras.

Remarque. Sur support numérique, les noms de matrice de ce paragraphe sont des liens hypertextes. Ils redirigent vers l'analyse les concernant si l'on clique dessus.

Structure de donnée.

Structure-A | Représentation en matrice creuse : les valeurs nulles sont implicites. C'est le cas de la totalité des matrices manipulées.

Structure-B | Représentation implicite de l'identité : $\phi[x, x] = 1$ est implicite. C'est le cas de : g^{t+1}, f^{t+1}, $g^{S,t+1}$, $f^{S,t+1}$.

Construction de matrices.

Construction-D | Construction d'une matrice qui ne dépend pas de son nombre de lignes. Cette construction est nécessaire pour le calcul de : χ, χ^B, χ^S, $f^{S,t+1}$, $g^{S,t+1}$, $h^{S,t+1}$.

Modification de matrices.

Modification-D | Ajout de lignes vides dans une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes. Cette modification est nécessaire pour le calcul de h^{t+1} ;

Modification-E | **Ajout de colonnes vides dans une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes**. Cette modification est nécessaire pour le calcul de : f^{t+1}, h^{t+1} ;

Modification-F | **Duplication d'une ligne d'une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes**. Cette modification est nécessaire pour le calcul de : ∂^{t+1}, g^{t+1} ;

Modification-G | **Duplication d'une colonne d'une matrice sans dépendre de son nombre de lignes ni de son nombre de colonnes**. Cette modification est nécessaire pour le calcul de : ∂^{t+1} ;

Modification-H | **Suppression de coefficients dans une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes**. Cette modification est nécessaire pour le calcul de : ∂^{t+1}.

Produit impliquant une matrice.

Produit-B | Produit matriciel $A \times B$ en parcourant les colonnes de B. Ce produit est nécessaire pour le calcul de : χ, χ^B, χ^S, $g^{S,t+1}$, $f^{S,t+1}$, $h^{S,t+1}$;

Produit-C | Produit d'une matrice avec un scalaire. Ce produit est nécessaire pour le calcul de : $g^{S,t+1}$, $h^{S,t+1}$, $f^{S,t+1}$.

Fusion de matrices.

Fusion-C | **Fusion de matrices** $A = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$**.** Cette fusion est nécessaire pour le calcul de : $\partial^{B,t+1}$, $\partial^{S,t+1}$, $f^{S,t+1}$, $g^{S,t+1}$, $h^{S,t+1}$.

Fusion-D | **Fusion de matrices** $A = \begin{bmatrix} A & B \end{bmatrix}$ Cette fusion est nécessaire pour le calcul de : g^{t+1}.

Fusion-E | **Fusion de matrices** $A = \begin{bmatrix} A \\ B \end{bmatrix}$ Cette fusion est nécessaire pour le calcul de : f^{t+1}.

2.1.5 Synthèse

L'application du théorème SECE aux opérations d'identification et de désidentification repose essentiellement sur deux calculs : le produit matriciel et la fusion de matrice. L'implémentation de ces deux calculs doit satisfaire des prérequis pour que le calcul de l'équivalence homologique γ^{t+1} par modification de l'équivalence homologique γ^t tire partie de l'hypothèse (ii). En fonction des processus de construction étudiés et des cas d'utilisation du théorème SECE, nous constatons que seule une partie des matrices de γ^t peut être maintenue afin de limiter les calculs. Nous distinguons deux cas d'utilisation du théorème SECE :

— le cas où l'on souhaite pouvoir calculer les générateurs d'homologie à n'importe quelle étape du processus de construction ;

— le cas où l'on peut se passer des générateurs d'homologie.

Nous distinguons trois types de processus de construction :

— ceux composés uniquement d'identifications ;

— ceux composés uniquement de désidentifications ;

— ceux composés d'identifications et de désidentifications.

Les matrices qui doivent être maintenues en fonction de ces différents cas sont répertoriées dans le tableau 2.3. En particulier, on observe qu'il est possible de limiter le nombre de matrices maintenues quand :

— on peut se passer des générateurs d'homologie et que le processus de construction étudié est composé uniquement d'identifications. Dans ce cas, les matrices f^t, h^t, $g^{S,t}$ et $h^{S,t}$ ne sont pas nécessaires ;

— le processus de construction est composé uniquement de désidentifications, que les générateurs d'homologie soient calculés ou non. Dans ce cas, les matrices $g^t, h^t, f^{S,t}, h^{S,t}$ ne sont pas nécessaires.

Enfin, nous constatons que les prérequis relevés se regroupent en plusieurs catégories :

— ceux concernant la **représentation matricielle**, avec par exemple la représentation implicite des valeurs nulles d'une matrice ;

— ceux concernant la **construction de matrices**, avec par exemple la construction d'une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes ;

— ceux concernant la **modification de matrices**, avec par exemple la suppression d'une ligne ou d'une colonne dans une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes ;

— ceux concernant le **produit matriciel**, avec par exemple le produit de deux matrices $A \times B$ en parcourant les colonnes de B ;

— ceux concernant la **fusion de matrices**, avec notamment deux fusions prédominantes,
  · dans le cas de l'identification, la modification d'une "grosse" matrice D par ajout de 3 "petites" matrices A, B, et C telles que $D = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$;
  · dans le cas de la désidentification, la modification d'une "grosse" matrice A par ajout de 3 "petites" matrices A, B, et C telles que $A = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$.

La totalité des prérequis d'implémentation s'applique aux matrices et aux opérations qui les accompagnent. Pour les satisfaire, les structures de données utilisées pour la représentation matricielle doivent donc être judicieusement choisies.

$Générateurs$	$Identifications$	$Désidentifications$	$Matrices$
Non	Oui	Non	$\partial^t, \partial^{B,t}, \partial^{S,t}, g^t, f^{S,t}$
Non	Non	Oui	$\partial^t, \partial^{B,t}, \partial^{S,t}, f^t, g^{S,t}$
Non	Oui	Oui	$\partial^t, \partial^{B,t}, \partial^{S,t}, f^t, g^t, h^t, f^{S,t}, g^{S,t}, h^{S,t}$
Oui	Oui	Non	$\partial^t, \partial^{B,t}, \partial^{S,t}, f^t, g^t, h^t, f^{S,t}, g^{S,t}, h^{S,t}$
Oui	Non	Oui	$\partial^t, \partial^{B,t}, \partial^{S,t}, f^t, g^{S,t}$
Oui	Oui	Oui	$\partial^t, \partial^{B,t}, \partial^{S,t}, f^t, g^t, h^t, f^{S,t}, g^{S,t}, h^{S,t}$

TABLE 2.3 – Les matrices de γ^t qu'il faut maintenir en fonction des cas d'utilisation du théorème SECE et des processus de construction étudiés. La colonne "Générateurs" indique si les générateurs d'homologie peuvent être calculés, les colonnes "Identifications" et "Désidentifications" indiquent si le processus de construction étudié est composé de ces opérations, et la colonne "Matrices" indiquent les matrices qui doivent être maintenues.

2.2 Matrices creuses

L'objectif de cette section est d'étudier les représentations matricielles existantes afin de vérifier s'il en existe une pouvant satisfaire les prérequis d'implémentation du théorème SECE soulevés dans la section 2.1. Dans un premier temps, nous nous intéressons aux formats standards de représentation de matrices creuses. Dans un second temps, nous étudions quelques bibliothèques logicielles proposant ces formats.

Remarque. Les prérequis d'implémentation sont récapitulés dans les paragraphes 2.1.2-c et 2.1.4-c.

2.2.1 Formats standards

Il existe des formats standards de représentation de matrices creuses. Pour les présenter, nous utilisons la matrice M suivante :

$$M = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{bmatrix} 2 & . & . \\ 3 & . & 5 \\ . & . & 4 \\ . & . & . \\ 1 & 6 & 7 \end{bmatrix} \end{array}$$

2.2.1-a Par coordonnées (COO).

Le format de représentation de matrices creuses par coordonnées [85] (COO, ou *Coordinate Format* en anglais) est relativement simple : une matrice creuse est entièrement représentée par ses valeurs non nulles, elles-mêmes représentées par un triplet (i, j, v) tel que :

— i correspond à la ligne de la valeur non nulle ;

— j correspond à sa colonne ;

— v correspond à sa valeur.

La matrice est alors représentée sous la forme de 3 tableaux :

— `rows`, dans lequel sont stockés les indices de lignes ;

— `cols`, dans lequel sont stockés les indices de colonnes ;

— `values`, dans lequel sont stockés les valeurs.

La figure 2.13 illustre la représentation de M en COO.

`rows`	0	1	2	1	4	4	4
`cols`	0	2	2	0	0	1	2
`values`	2	5	4	3	1	6	7

FIGURE 2.13 – Les 3 tableaux `rows`, `cols` et `values` de la représentation de M en COO.

Remarque. Pour les besoins de ce paragraphe, nous omettons volontairement de considérer le fait que les éléments de ces tableaux puissent être triés ou non. En effet, les deux versions sont acceptables pour le format COO. Choisir l'une ou l'autre fait sens lorsque le format s'accompagne d'opérations telles que l'accès à un élément ou l'insertion d'un nouvel élément. L'objectif du paragraphe étant simplement de présenter les formats, nous ne descendons pas à ce niveau de discours, qui dépend par ailleurs des implémentations du format dans les bibliothèques logicielles, et des besoins auxquels elles répondent.

2.2.1-b Compression de lignes (CSR).

Le format de représentation de matrices par compression de lignes (CSR, ou *Compressed Sparse Rows* en anglais) reprend l'idée du format COO, en "compressant" le tableau `rows`, c'est-à-dire que les indices de lignes ne sont pas tous explicitement représentés. Intuitivement, cela consiste à considérer la matrice non plus comme un ensemble de valeurs non nulles, mais comme un ensemble de vecteurs "lignes", eux-mêmes composés de valeurs non nulles. La matrice est représentée sous la forme de 3 tableaux :

- `cols`, dans lequel sont stockés les indices de colonnes des valeurs non nulles ;
- `values`, dans lequel sont stockées les valeurs ;
- `rows-first-idx`, qui contient une entrée pour chaque ligne de M. La valeur de son x-ème élément est l'indice de la première valeur non nulle de la ligne x dans les tableaux `cols` et `values`. Un élément supplémentaire est ajouté à la fin de `rows-first-idx`, ayant pour valeur le nombre de valeurs non nulles de M (donc la taille des tableaux `cols` et `values`).

Soit (i, j, v) une valeur non nulle. En CSR :

- j est explicitement représenté dans `cols` ;
- v est explicitement représenté dans `values` ;
- i n'est pas explicitement représenté mais peut être déduit. Soit x l'indice de `values` contenant v. La valeur de i est telle que `rows-first-idx[i]` $\leq x <$ `rows-first-idx[i+1]`.

En CSR, la matrice peut être lue valeur non nulle par valeur non nulle mais aussi ligne par ligne. La figure 2.14 illustre la représentation de M en CSR.

Remarque. Observons que ce format nécessite que les tableaux `cols` et `values` soient triés par lignes (mais une ligne n'est pas forcément triée).

Exemple. Considérons la représentation de la figure 2.14. On observe en particulier que :

- la ligne 3 ne contient aucune valeur non nulle car `rows-first-idx[4]` $-$ `rows-first-idx[3]` $= 4 - 4 = 0$;
- la ligne 1 contient deux valeurs non nulles car `rows-first-idx[2]` $-$ `rows-first-idx[1]` $= 3 - 1 = 2$. L'indice de sa première valeur non nulle dans `cols` ou `values` est `rows-first-idx[1]` $= 1$. Elle correspond à $(1, 0, 3)$ car `cols[1] = 0` et `values[1] = 3`

Cherchons maintenant à retrouver un indice de ligne i et de colonne j en partant d'une valeur non nulle v. Considérons :

- l'élément de `values` à l'indice $x = 3$. On a $v = 4$ car `values[3] = 4`. L'indice de colonne j de cette valeur non nulle est la valeur du $x-ème$ élément de `cols`, c'est-à-dire `cols[3] = 2` d'où $j = 2$. Enfin, l'indice de ligne est $i = 2$ car `rows-first-idx[2]` $= 3 \leq x =$

x	0	1	2	3	4	5	6
`rows-first-idx`	0	1	3	4	4	*7*	
`cols`	0	0	2	2	0	1	2
`values`	2	3	5	4	1	6	7

FIGURE 2.14 – Les 3 tableaux `rows-first-idx`, `cols` et `values` de la représentation de M en CSR. L'élément supplémentaire de `rows-first-idx` est en italique. La ligne x sert simplement d'aide à la lecture et ne fait pas partie du format. Elle représente les indices des éléments des tableaux.

$3 <$ `rows-first-idx[3]` $= 4$. La valeur non nulle qui correspond à l'élément d'indice $x = 3$ dans `values` est donc $(2, 2, 4)$;

— l'élément de `values` à l'indice $x = 6$. On a $v = 7$ car `values[6] = 7`. L'indice de colonne est $j = 2$ car `cols[6] = 2`. Enfin, l'indice de ligne est $i = 4$ car `rows-first-idx[4]` $= 4 \leq x = 6 <$ `rows-first-idx[5]` $= 7$. La valeur non nulle qui correspond à l'élément d'indice $x = 6$ dans `values` est donc $(4, 2, 7)$.

2.2.1-c Compression de colonnes (CSC)

Le format de représentation par compression de colonnes (CSC, ou *Compressed Sparse Columns* en anglais) reprend la logique du format CSR, en compressant cette fois-ci le tableau `cols`. La matrice est représentée sous la forme de 3 tableaux :

— `rows`, dans lequel sont stockés les indices de lignes des valeurs non nulles ;

— `values`, dans lequel sont stockées les valeurs non nulles ;

— `cols-first-idx`, qui contient une entrée pour chaque colonne de M. La valeur de son x-ème élément est l'indice de la première valeur non nulle de la colonne x dans les tableaux `rows` et `values`. Un élément supplémentaire est ajouté à la fin de `cols-first-idx`, ayant pour valeur le nombre de valeurs non nulles de M (donc la taille des tableaux `rows` et `values`).

Soit (i, j, v) une valeur non nulle. En CSC :

— i est explicitement représenté dans `rows` ;

— v est explicitement représenté dans `values` ;

— j n'est pas explicitement représenté mais peut être déduit. Soit x l'indice de `values` contenant v. La valeur de j est telle que `cols-first-idx[j]` $\leq x <$ `cols-first-idx[j+1]`.

En CSC, la matrice peut être lue valeur non nulle par valeur non nulle mais aussi colonne par colonne. La figure 2.15 illustre la représentation de M en CSC.

Remarque. Observons que ce format nécessite que les tableaux `rows` et `values` soient triés par colonnes (mais une colonne n'est pas forcément triée).

Exemple. Considérons la représentation de la figure 2.15. On observe en particulier que :

x	0	1	2	3	4	5	6
`cols-first-idx`	0	3	4	*7*			
`rows`	0	1	4	4	1	2	4
`values`	2	3	1	6	5	4	7

FIGURE 2.15 – Les 3 tableaux `cols-first-idx`, `rows` et `values` de la représentation de M en CSC. L'élément supplémentaire de `cols-first-idx` est en italique. La ligne x sert simplement d'aide à la lecture et ne fait pas partie du format. Elle représente les indices des éléments des tableaux.

— la colonne 1 ne contient qu'une valeur non nulle car `cols-first-idx[2]`$-$`cols-first-idx[1]` $= 4-3=1$;

— la colonne 0 contient trois valeurs non nulles car `cols-first-idx[1]`$-$`cols-first-idx[0]` $= 3-0=3$. L'indice de sa première valeur non nulle dans `rows` ou `values` est `cols-first-idx[0]` $= 0$. Elle correspond à $(0,0,2)$ car `rows[0] = 0` et `values[0] = 2`.

Cherchons maintenant à retrouver un indice de ligne i et de colonne j en partant d'une valeur non nulle v. Considérons :

— l'élément de `values` à l'indice $x=3$. On a $v=6$ car `values[3] = 6`. L'indice de ligne i de cette valeur non nulle est la valeur du $x-$ème élément de `rows`, c'est-à-dire `rows[3] = 4` d'où $i=4$. Enfin, l'indice de colonne est $j=1$ car `cols-first-idx[1]` $= 3 \leq x = 3 <$ `cols-first-idx[2]` $= 4$. La valeur non nulle qui correspond à l'élément d'indice $x=3$ dans `values` est donc $(4,1,6)$;

— l'élément de `values` à l'indice $x=6$. On a $v=7$ car `values[6] = 7`. L'indice de ligne est $i=4$ car `rows[6] = 4`. Enfin, l'indice de colonne est $j=2$ car `cols-first-idx[2]` $= 4 \leq x = 6 <$ `cols-first-idx[3]` $= 7$. La valeur non nulle qui correspond à l'élément d'indice $x=6$ dans `values` est donc $(4,2,7)$.

2.2.1-d Compression de lignes par bloc (BSR).

Le format de représentation par blocs, avec compression de lignes (BSR, ou *Block Compressed Row Format* en anglais), reprend la logique du format CSR à cela près qu'il ne représente non plus directement les valeurs non nulles, mais une décomposition de la matrice M en sous-matrices $M_{x,y}$ non nulles, toutes de même taille, appelées "blocs" et où :

— x désigne l'indice de ligne de $M_{x,y}$ dans la décomposition de M en sous-matrices ;

— y désigne son indice de colonne.

La matrice M est représentée sous la forme de 3 tableaux :

— `cols`, dans lequel sont stockés les indices de colonnes des blocs de la décomposition ;

— `values`, dans lequel sont stockées les valeurs des blocs de la décomposition ;

— `rows-first-idx`, qui contient une entrée pour chaque ligne de la décomposition de M en sous-matrices. La valeur de son x-ème élément est l'indice du premier bloc de la ligne x dans les tableaux `cols` et `values`. Un élément supplémentaire est ajouté à la fin de ce tableau, ayant pour valeur le nombre de blocs de M (donc la taille du tableau `cols`).

Soit $M_{x,y}$ un bloc non nulle. En BSR :

— y est explicitement représenté dans `cols`. La taille de `cols` correspond au nombre de blocs non nuls ;

— les valeurs de $M_{x,y}$ sont explicitement représentées dans `values`. La taille de `values` est un multiple de la taille de `cols` qui dépend de la taille des blocs choisie ;

— x n'est pas explicitement représenté, mais peut être déduit. Soit λ l'indice de `cols` contenant y. La valeur de x est telle que `rows-first-idx[x]` $\leq \lambda <$ `rows-first-idx[x+1]`.

La figure 2.16b illustre la représentation de M en BSR.

Remarque. Notons que par rapport à des matrices creuses "classiques", l'optimisation se fait sur les blocs nuls et non sur les valeurs nulles. Dit autrement, certains 0 peuvent être représentés, c'est par exemple le cas de $M[0,1]$ qui est représenté dans $M_{0,0}$, dans `values`. De plus, si le nombre de lignes de chaque bloc n'est pas un multiple du nombre de lignes de M, alors le bloc est rempli par ajout de 0 "fictifs". Il en est de même pour les colonnes. La taille de `values` est donc celle de `cols` multipliée par le nombre de valeurs par blocs. Ce format est intéressant pour des matrices de grandes dimensions et où une décomposition en blocs de grande taille garantie un nombre important de blocs nuls. Plus les valeurs non nulles sont concentrées en un même endroit, plus ce format est intéressant.

$$\begin{bmatrix} M_{0,0} & M_{0,1} \\ M_{1,0} & M_{1,1} \end{bmatrix}$$

(a)

λ	0	1	2
`rows-first-idx`	0	2	*3*
`cols`	0	1	1
`values`	$M_{0,0}$	$M_{0,1}$	$M_{1,1}$
	2 0 3 0	0 0 5 0	4 0 0 0

(b)

FIGURE 2.16 – La représentation de M en BSR avec une taille de bloc 2×2. La ligne λ sert simplement d'aide à la lecture et ne fait pas partie du format. Elle représente l'indice des éléments de `rows-first-idx` et `cols`. (a) La décomposition. (b) Les 3 tableaux `rows-first-idx`, `cols` et `values` de la représentation de M en BSR. En italique, l'élément supplémentaire de `rows-first-idx`.

Exemple. Considérons la représentation de la figure 2.16, dans laquelle la matrice M est décomposée en 4 blocs de taille 2×2. D'abord, observons que le seul bloc représenté implicitement avec cette taille de bloc est $M_{1,0}$. Ensuite, observons que :

— la ligne 0 de la décomposition contient 2 blocs non nuls car `rows-first-idx[1]` $-$ `rows-first-idx[0]` $= 2 - 0 = 2$;

— la ligne 1 de la décomposition contient 1 bloc non nul car `rows-first-idx[2]` $-$ `rows-first-idx[1]` $= 3 - 2 = 1$.

Cherchons maintenant à retrouver un indice de ligne i et de colonne j en partant d'une valeur non nulle v. Considérons l'élément de `values` d'indice 6, du bloc $M_{0,1}$. On a $v = 5$ car `values[6]` = 5. Pour calculer j, on :

— calcule λ pour $v = 5$. Il y a 4 éléments par bloc, donc $\lambda = 6/4 = 1$;

— calcule l'indice de colonne $j_{M_{0,1}}$ de la valeur 5 dans $M_{0,1}$: $j_{M_{0,1}} = 6 \mod b_x = 6 \mod 2$, où b_x désigne la largeur des blocs ;

— calcule $j = cols[\lambda] * b_x + j_{M_{0,1}} = 1 * 2 + 0 = 2$. Le produit $cols[\lambda] * b_x$ représente le décalage de la colonne 0 de $M_{0,1}$ par rapport à la colonne 0 de M.

Pour calculer i, on :

— cherche x tel que `rows-first-idx[x]` $\leq \lambda = 1 <$ `rows-first-idx[x+1]`. On trouve $x = 0$;

— calcule l'indice de ligne $i_{M_{0,1}}$ de la valeur 5 dans $M_{0,1}$: $i_{M_{0,1}} = (6 \mod 4)/b_y = 2/2 = 1$, où b_y désigne la hauteur des blocs ;

— calcule $i = x * b_y + i_{M_{0,1}} = 0 + 1 = 1$. Le produit $x * b_y$ représente le décalage de la ligne 0 de $M_{0,1}$ par rapport à la ligne 0 de M.

La valeur non nulle correspondant à la case 6 de `values` est donc $(1, 2, 5)$.

2.2.2 Bibliothèques logicielles

Dans ce paragraphe, nous étudions différentes bibliothèques logicielles pour la représentation de matrices creuses, dans le but d'en trouver une avec laquelle satisfaire l'ensemble des prérequis d'implémentation issus des sections 2.1.2-c, 2.1.3-c et 2.1.4-c. Les bibliothèques logicielles considérées sont :

— *Eigen*, développée par l'INRIA. Elle propose une implémentation des formats CSR et CSC ;

— *Scipy*, un projet communautaire développé par des *contributeurs* de différentes institutions, et financé par divers sponsors dont NumFOCUS et Tidelift. Elle propose une implémentation des formats COO, CSR/CSC et BSR ;

— *IntelMKL*, développée par Intel. Elle propose une implémentation des formats COO, CSR/CSC et BSR ;

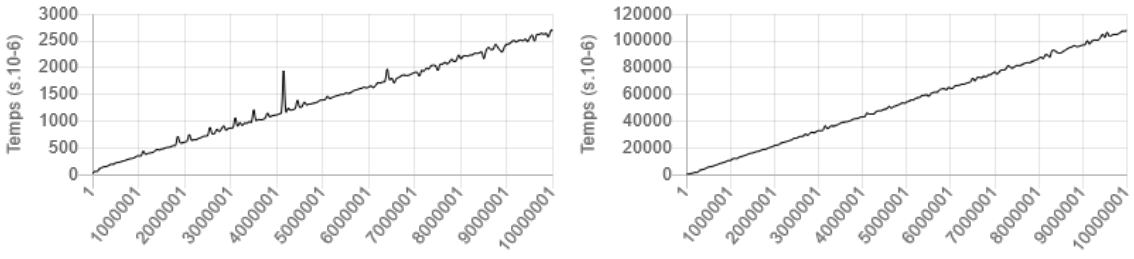
— *cuSPARSE*, une partie de *Cuda*, développée par Nvidia. Elle propose une implémentation des formats COO, CSR/CSC et BSR. Elle a particularité de permettre le développement d'algorithmes sur processeur graphique Nvidia.

Remarque. Les mesures de ce paragraphe ont été prises sur une machine équipée d'un Intel i3-8100 @3.60GHz, de 16Go de RAM à 2400MHz et d'un processeur graphique Geforce GTX 750. Pour alléger le paragraphe, nous faisons le choix de ne présenter qu'un prérequis qui pose problème pour une bibliothèque logicielle donnée.

2.2.2-a Eigen.

Considérons le prérequis Modification-A, "*Suppression d'une ligne d'une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes*". Les fonctions de modifications de matrices creuses d'Eigen sont relativement limitées du fait des formats implémentés. Plus précisément, si la bibliothèque offre de nombreuses fonctions de lecture pour ses matrices creuses (dont une lecture par bloc), elle fait le choix de limiter les fonctions d'écriture à celles qui peuvent être implémentées de manière efficace au vu des formats proposés. Pour autant, la suppression

de ligne reste possible avec Eigen, de manière détournée. Pour cela, on permute la ligne à supprimer sur une extrémité [9] de la matrice, puis on change ses dimensions en lui enlevant une ligne.



(a) CSR (b) CSC

FIGURE 2.17 – Évolution du temps (en microsecondes) en fonction de n pour la suppression d'une ligne dans une matrice de dimension m par n, avec Eigen, pour $m = 1000$ et n variable compris entre 1 et $10\,000\,001$.

Pour tester ce fonctionnement, nous avons choisi de construire une matrice M de dimension m par n avec m fixée à 1000 et n variable entre 1 et $10\,000\,001$, et composée d'au plus 2 valeurs non nulles par ligne. Pour chaque valeur de n, on supprime une ligne arbitraire, en l'occurence la ligne 3, de M. Plus précisément, nous utilisons :

— une matrice de permutation P créée avec `PermutationMatrix<Dynamic, Dynamic, int> P(m)`, de dimension m par m. Elle est initialisée comme étant l'identité, puis modifiée de sorte à ce que les lignes 3 et $m-1$ de M soient interchangées ;

— la fonction `conservativeResize` pour redimensionner M après la permutation, elle même réalisée avec un produit $M = P * M$.

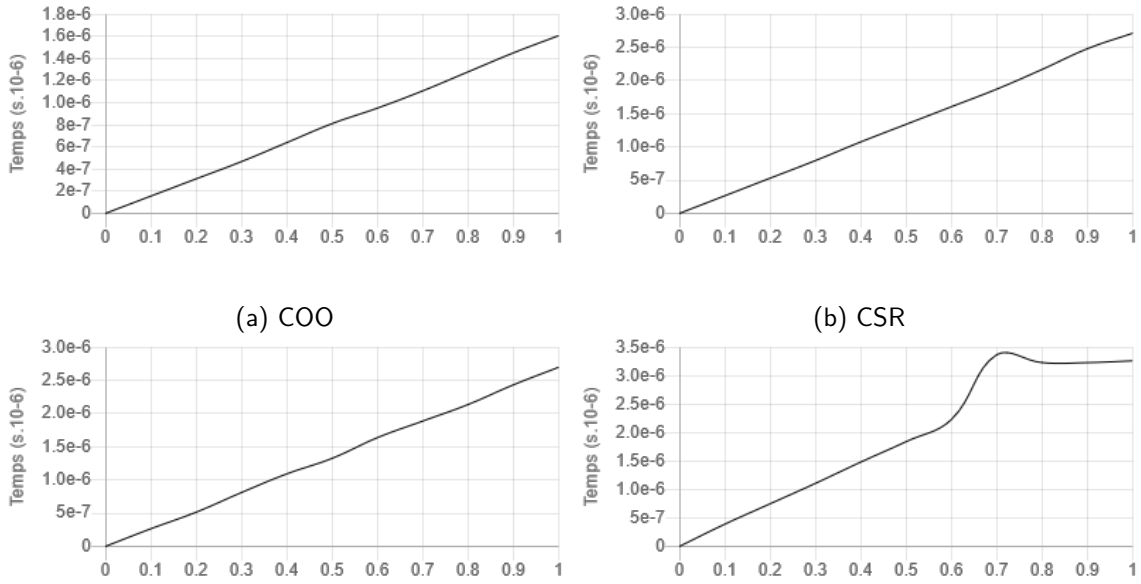
Les temps mesurés sont illustrés sur les graphiques de la figure 2.17b. En particulier, observons qu'avec les deux formats, la complexité de la suppression de ligne est linéaire en fonction du nombre de colonnes de M. En conclusion, le prérequis Modification-A ne peut pas être satisfait avec Eigen, et, plus généralement, **tous les prérequis d'implémentation ne sont pas satisfaits avec Eigen**.

2.2.2-b SciPy.

Considérons le prérequis Fusion-A, "*Fusion de matrices* $D = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$". Scipy propose une fonction particulièrement utile pour cela : `bmat`. Elle permet de construire une matrice creuse à partir de plusieurs sous-matrices. Pour la tester, nous construisons une matrice D de dimension m par n, avec m et n fixées à 10000. Les valeurs non nulles de D sont tirées aléatoirement, avec une densité variable entre 0 et 1 (à 1, D est une matrice dense). Nous utilisons également une matrice B de dimension 1 par n, une matrice C de dimension m par 1, et une matrice A de dimension 1 par 1. Pour A,B et C, la case à l'intersection de la ligne 0 et de la colonne 0 est l'unique valeur non nulle. Les temps mesurés sont illustrés sur la figure 2.18.

On observe qu'avec tous les formats, la complexité en temps est linéaire en fonction de la densité de D. Dit autrement, le nombre de valeurs non nulles de D joue un rôle dans le temps

9. Pour supprimer une ligne, on la permute en bas. Pour supprimer une colonne, on la permute à droite.



(a) COO (b) CSR

(c) CSC (d) BSR

FIGURE 2.18 – Évolution du temps (en microsecondes) de la fonction `bmat` de Scipy en fonction de la densité de D pour une fusion de matrice du type Fusion-A.

de calcul de `bmat`, c'est-à-dire que cette fusion ne dépend pas uniquement des caractéristiques des 3 matrices fusionnées A, B et C. En conséquence, **tous les prérequis d'implémentation ne sont pas satisfaits avec SciPy**.

2.2.2-c IntelMKL.

Intel Math Kernel Library se décompose en plusieurs grandes parties, dont l'une est dédiée aux matrices creuses. Son fonctionnement est particulier : c'est un "noyau" avec lequel on interagit uniquement au travers des fonctions proposées. Par exemple, pour "créer" une matrice au format COO, on utilise la fonction `mkl_sparse_?_create_coo` [10] en créant au préalable les tableaux `rows`, `cols` et `values` que l'on donne en paramètre à la fonction. La matrice est alors connue par le noyau, et représentée dans un format "interne" au noyau. Elle peut être utilisée en paramètre d'autres fonctions. Parmi les fonctions proposées, aucune ne permet de fusionner des matrices déjà connues par le noyau. De plus, à notre connaissance, il n'est pas possible d'implémenter une surcouche qui le fasse en respectant les prérequis d'implémentation. Une fonction avec laquelle il est possible de modifier une matrice du noyau est `mkl_sparse_?_set_value`. Elle permet de changer une valeur non nulle déjà existante (cf. [86], page 294). Il n'est donc pas possible d'ajouter de nouvelles valeurs non nulles à une matrice du noyau de cette façon. En conséquence, **tous les prérequis d'implémentation ne sont pas satisfaits avec IntelMKL**.

10. IntelMKL permet de choisir le type des éléments d'une matrice creuse parmi 4 types. Le point d'interrogation dans le nom des fonctions indique qu'elle est implémentée pour chacun de ces types ; il suffit de le remplacer par le type que l'on utilise (par exemple s pour des flottants simple précision).

2.2.2-d cuSPARSE.

Cette bibliothèque est une partie de CUDA, et sert essentiellement à l'implémentation d'algorithmes parallèles sur processeur graphique NVIDIA. Son principe est sensiblement le même qu'IntelMKL : c'est une boîte noire avec laquelle on interagit uniquement au travers des fonctions proposées. Par exemple, pour créer une matrice au format COO, on utilise la fonction `cusparseCreateCoo` en créant au préalable les tableaux les tableaux `rows`, `cols` et `values` que l'on donne en paramètre à la fonction. La matrice est alors connue par la boîte noire et est utilisable pour d'autres calculs. Parmi les fonctions proposées, aucune ne permet de fusionner des matrices. De plus, à notre connaissance, il n'est pas possible d'implémenter une surcouche qui le fasse. Une fonction avec laquelle il est possible de modifier une matrice de la boîte noire est `cusparseSpMatSetValues`. Elle permet de modifier toutes les valeurs non nulles déjà existantes, mais n'en crée pas de nouvelles. Il n'est donc pas possible d'ajouter de nouvelles valeurs non nulles à une matrice de la boîte noire de cette façon. En conséquence, **tous les prérequis d'implémentation ne sont pas satisfaits avec cuSparse**.

2.2.3 Conclusion

Au bout du compte, on constate que la plupart des bibliothèques de matrices creuses sont limitées en écriture, ce qui pose problème notamment pour la fusion de matrices. Ces limitations s'expliquent en partie par le fait que ces bibliothèques sont avant tout pensées pour le calcul (produit, résolution de systèmes linéaires, calcul de valeurs propres etc...). Elles ne se soucient pas forcément de "l'évolution" des matrices. Aucune des bibliothèques logicielles essayées ne satisfait l'ensemble des prérequis d'implémentation du théorème SECE des paragraphes 2.1.2-c, 2.1.3-c et 2.1.4-c. Par ailleurs, notons que la limitation en écriture de ces bibliothèques fait sens car elle découle directement des formats standards de représentation de matrices. Pour cette raison, nous avons choisi de concevoir notre propre représentation matricielle.

2.3 Implémentation

Pour implémenter le théorème SECE en respectant les prérequis des paragraphes 2.1.3-c et 2.1.4-c, nous avons conçu et développé notre propre représentation matricielle. Elle fonctionne de pair avec des *répertoires de cellules*. Cette section est composée de plusieurs sous-sections :

— en 2.3.1, nous présentons les structures de données utilisées pour la représentation matricielle et les répertoires de cellules ;

— en 2.3.2, nous montrons en quoi ces structures de données permettent de vérifier les prérequis d'implémentation des paragraphes 2.1.3-c et 2.1.4-c ;

— en 2.3.3, nous donnons un exemple d'utilisation de cette représentation matricielle et du répertoire de cellules dans le cas de l'identification ;

— en 2.3.4, nous présentons l'implémentation en C++ que nous avons réalisé.

2.3.1 Structures de données

Nous présentons les structures de données utilisées pour la représentation matricielle, les répertoires de cellules et pour la représentation de l'identification.

2.3.1-a Répertoire de cellules

Pour rappel, les complexes de chaînes que nous manipulons sont systématiquement engendrés par des structures topologiques dont les cellules forment une base des chaînes. Un répertoire de cellules est une représentation d'une telle base. Il suit son évolution au fil d'un processus de construction. Une base évolue lorsque :

— une cellule meurt. C'est par exemple le cas si une cellule ne survit pas à une identification ;

— une cellule est créée. C'est par exemple le cas lorsqu'une cellule est éclatée au cours d'une désidentification.

Un répertoire de cellules se décompose en plusieurs parties :

— une collection représentant la base de cellules. Nous avons choisi d'utiliser un `deque` pour représenter cette collection ;

— une collection représentant l'ensemble des matrices dans lesquelles la base représentée est le domaine et/ou le codomaine. Nous avons choisi d'utiliser une `liste` pour représenter cette collection.

Soit $B_X = \{\sigma_0, \ldots, \sigma_q\}$ une base de cellules représentée par un répertoire composé d'un `deque` D_{B_X} et d'une `liste` L_{B_X}.

Base de cellules. À chaque cellule de B_X est attribué un indice unique au fil du processus de construction. Le i-ème élément de D_{B_X} est un triplet $(isAlive, prev, next)$ où :

— $isAlive$ est un booléen indiquant l'état de la i-ème cellule. Il vaut vrai si la i-ème cellule est en vie, faux sinon ;

— $prev$ est un entier représentant l'indice de la cellule en vie qui précède la i-ème cellule. Nous utilisons le symbole $-$ pour indiquer qu'aucune cellule ne précède la i-ème cellule ;

— $next$ est un entier représentant l'indice de la cellule en vie qui suit la i-ème cellule. Nous utilisons le symbole $-$ pour indiquer qu'aucune cellule ne suit la i-ème cellule ;

Lorsqu'une cellule meurt, elle n'appartient plus à B_X mais elle induit toujours un élément dans D_{B_X}, dont le champ $isAlive$ vaut faux. Le nombre d'éléments de D_{B_X} ne fait qu'augmenter au fil du processus de construction. Une cellule meurt dans deux cas :

— lorsqu'elle est non-survivante à une identification ;

— lorsqu'elle est "réduite". Soit $\rho : (C, \partial) \Rightarrow (C', \partial')$ une réduction. Une cellule réduite est une cellule qui existe dans (C, ∂) et n'existe pas dans (C', ∂'). Par exemple, sur la figure 2.22, la cellule f_0 de $(C^{B,t}, \partial^{B,t})$ est réduite car elle n'existe pas dans $(C^{S,t}, \partial^{S,t})$.

L'indice de la première cellule vivante d'une base est maintenu tout au long du processus de construction et est représenté par un entier `first`.

Remarque. Pour simplifier, nous désignons les cellules par leur indice par la suite.

Ensemble des matrices. Les éléments de L_{B_X} sont des pointeurs sur des matrices ayant B_X pour domaine et/ou codomaine. Cette liste est utilisée pour mettre à jour les matrices concernées lorsque B_X évolue, c'est-à-dire lorsqu'une cellule est créée ou lorsqu'une cellule meurt.

2.3.1-b Représentation matricielle

Pour rappel, les matrices que nous manipulons sont parfois étudiées comme des variations de l'identité (cf. section 2.1). Toutes les matrices que nous manipulons se décomposent en plusieurs parties :

— un domaine et un codomaine, chacun représenté par un répertoire de cellules (potentiellement le même) ;

— un booléen $isIdentityVariation$, valant vrai lorsqu'une matrice est étudiée comme une variation de l'identité ;

— des coefficients.

Soit une matrice $M : B_X \to B_Y$ où B_X et B_Y sont des bases représentées par des répertoires de cellules composés respectivement d'un `deque` D_{B_X} et d'une `liste` L_{B_X}, et d'un `deque` D_{B_Y} et d'une liste L_{B_Y}.

Représentation des coefficients. Pour représenter un coefficient de M, nous utilisons un triplet (i, j, v), où :

— i est un entier qui désigne son indice de ligne ;

— j est un entier qui désigne son indice de colonne ;

— v est un entier qui désigne sa valeur.

La i-ème ligne de M représente l'image de la i-ème cellule de B_X par M. La j-ème colonne de M représente les antécédents de la j-ème cellule de B_Y par M. Le coefficient $M[i = j, i = j] = 1$ est ajouté à l'image ou aux antécédents d'une cellule par M si $isIdentityVariation$ vaut vrai et que la i-ème cellule est vivante dans D_{B_X} et D_{B_Y}. Les coefficients sont des maillons reliés entre eux par un chaînage double horizontal et un chaînage double vertical. Les voisins du haut, de gauche, du bas et de droite d'un maillon sont respectivement nommés `top`, `left`, `bottom` et `right`.

Remarque. Comme nous le verrons dans le paragraphe 2.3.2-e, la représentation matricielle peut être utilisée de deux façons : avec des lignes/colonnes indicées négativement ou non. Le chaînage

double entre maillons n'est nécessaire que lorsqu'on autorise des indices négatifs. Un chaînage simple est suffisant dans l'autre cas. Cela est discuté plus en détail dans le paragraphe 2.3.2-e.

Points d'entrée. L'accès au premier ou au dernier maillon d'une ligne ou d'une colonne se fait un travers d'un *point d'entrée*. Nous distinguons :

— les matrices **maintenues** tout au long d'un processus de construction, construites au début de celui-ci. Leurs points d'entrée sont stockés dans des `deque`. Lorsqu'une cellule meurt, les lignes et/ou colonnes qui la représente sont vidées, c'est-à-dire que les maillons sont supprimés mais il existe toujours un élément dans les `deque` pour ces lignes/colonnes. Lorsqu'une cellule est créée, un élément est ajouté aux `deque` contenant les points d'entrée. En particulier, notons que la construction d'une telle matrice dépend de son nombre de points d'entrée, et la complexité en temps de l'accès à un point d'entrée est en temps constant ;

— les matrices **temporaires**, construites pour le passage d'une étape de construction à la suivante, puis détruites. Leurs points d'entrée sont stockés dans des `dictionnaires`. Lorsqu'une cellule meurt, les lignes/colonnes qui la représente sont supprimées, c'est-à-dire qu'en plus des maillons, des éléments des `dictionnaires` sont supprimés. Lorsqu'une cellule est créée, aucun point d'entrée n'est créé. En particulier, notons que la construction d'une telle matrice est en temps constant, et la complexité en temps de l'accès à un point d'entrée est logarithmique et dépend de la taille des `dictionnaires`.

Remarque. Pour rappel, la complexité en temps de la suppression d'un élément dans un `dictionnaire` est logarithmique en fonction de son nombre d'éléments. Comme nous le verrons dans la section 2.3.2, les opérations de suppression de lignes ou de colonnes ne sont jamais appliquées à des matrices temporaires dans notre cas.

Exemple. Soient les bases de cellules $B_X = \{0, 2, 3\}$ et $B_Y = \{0, 2\}$, et les matrices $M_0 : B_X \to B_X$, maintenue, et $M_1 : B_X \to B_Y$, temporaire. On suppose que la cellule 1 de B_Y est morte et que $isIdentityVariation$ vaut vrai pour M_1. Les matrices M_0 et M_1 sont :

$$M_0 = \begin{array}{c} \\ 0 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 2 & 3 \end{array} \\ \begin{bmatrix} 1 & . & -5 \\ . & . & 2 \\ . & 1 & . \end{bmatrix} \end{array} \qquad M_1 = \begin{array}{c} \\ 0 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 2 \end{array} \\ \begin{bmatrix} . & . \\ . & . \\ . & 2 \end{bmatrix} \end{array}$$

Les répertoires de cellules représentant B_X et B_Y sont :

	D_{B_X}	L_{B_X}
0	$(vrai, -, 2)$	M_0
1	$(faux, -, -)$	M_1
2	$(vrai, 0, 3)$	
3	$(vrai, 2, -)$	
`first`	0	

	D_{B_Y}	L_{B_Y}
0	$(vrai, -, 2)$	M_1
1	$(faux, -, -)$	
2	$(vrai, 0, -)$	
`first`	0	

Pour rappel, le symbole $-$ indique qu'aucune cellule vivante ne précède ou ne suit la cellule considérée. La représentation de M_0 est illustrée sur la figure 2.19 et celle de M_1 est illustrée sur la figure 2.20. Observons que :

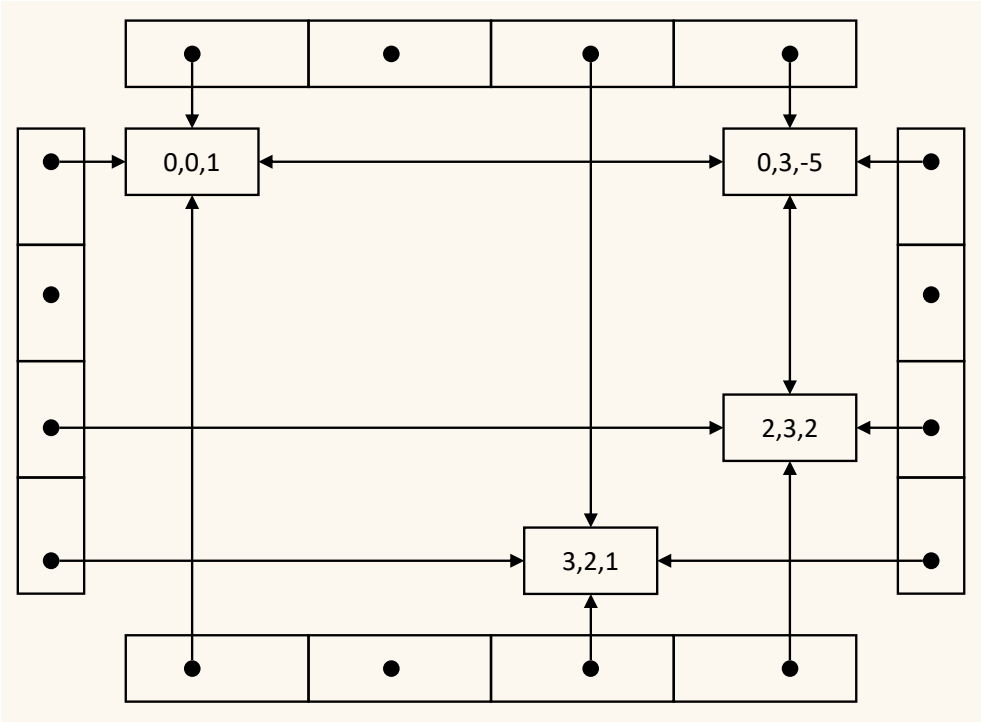


FIGURE 2.19 – Représentation d'une matrice $M_0 : B_X \to B_X$ avec $B_X = \{0, 2, 3\}$. Ses points d'entrées sont stockés dans des `deque`.

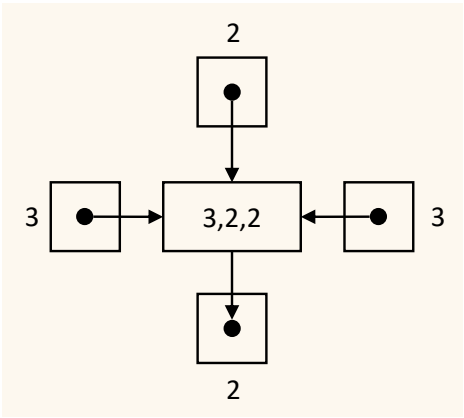


FIGURE 2.20 – Représentation d'une matrice $M_1 : B_X \to B_Y$ avec $B_X = \{0, 2, 3\}$ et $B_Y = \{0, 2\}$. Ses points d'entrée sont stockés dans des `dictionnaires`. Les labels des points d'entrée correspondent aux clefs des `dictionnaires`.

— les points d'entrée de M_0 sont stockés dans des `deque`. Notons que la cellule d'indice 1, morte, induit un point d'entrée dans ces `deque` ;

— les points d'entrée de M_1 sont stockés dans des `dictionnaires`. Notons que seules la ligne 3 et la colonne 2 induisent un point d'entrée dans ces `dictionnaires`.

Enfin, notons que $M_1[0, 0] = 1$ et $M_1[2, 2] = 1$ sont des coefficients implicites de M_1 car :

— $isIdentityVariation$ vaut vrai pour M_1 ;

— la cellule d'indice 0 est vivante dans D_{B_X} et dans D_{B_Y} ;

— la cellule d'indice 2 est vivante dans D_{B_X} et dans D_{B_Y}.

2.3.1-c Représentation de l'identification

Soit un complexe de chaînes $(C', \partial')^{n\in\mathbb{N}}$ et une identification caractérisée par $I = (I^p)^{0\leq p\leq n}$ et $\zeta : B_{C'} \to B_{C'}$. Pour représenter l'identification, nous avons choisi de ne représenter que les ensembles de cellules composés d'au moins deux cellules. Dit autrement, toute cellule qui n'est pas identifiée avec au moins une autre cellule n'apparaît pas dans la représentation de l'identification. Plus précisément :

— I^p est représentée par un `dictionnaire` S_{I^p} tel que :
  · une clef est l'indice d'une p-cellule σ survivante, dans la base des p-chaînes, et identifiée avec au moins une autre cellule ;
  · une valeur est un `tableau` contenant les indices des cellules non-survivantes identifiées avec σ.

— I est représentée par un `dictionnaire` S_I tel que :
  · une clef est une dimension p dans laquelle au moins une cellule est identifiée avec une autre ;
  · une valeur est S_{I^p}.

Enfin, on suppose que les parcours de S_I ou des différents S_{I^p} sont déterministes, c'est-à-dire que tant qu'ils ne sont pas modifiés, leurs éléments sont toujours parcourus dans le même ordre.

Remarque. Selon l'implémentation du `dictionnaire`, les différents S_{I^p} peuvent être plus simplement vu comme des structures de donnée de type *Union-Find*.

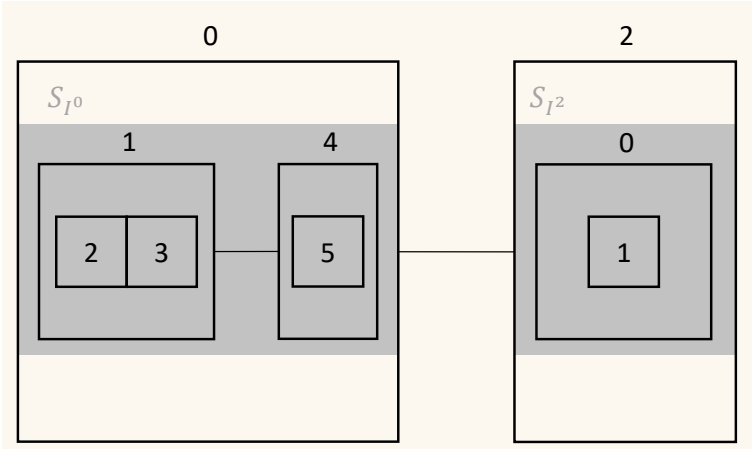


FIGURE 2.21 – Une identification représentée par un `dictionnaire` S_I dont les clefs sont 0 et 2 et dont les valeurs sont, respectivement, S_{I^0} et S_{I^2}. S_{I^0} est un `dictionnaire` dont les clefs sont 1 et 4 et dont les valeurs sont, respectivement, un `tableau` de taille 2 contenant 2 et 3, et un `tableau` de taille 1 contenant 5. S_{I^2} est un `dictionnaire` dont la clef est 0 et la valeur est un `tableau` de taille 1 contenant 1. Les deux dictionnaires S_{I^0} et S_{I^2} sont mis en évidence par un fond gris.

Exemple. L'identification illustrée sur la figure 2.21 représente une identification caractérisée par $I^0 = \{\{1, 2, 3\}, \{4, 5\}\}$ et $I^2 = \{\{0, 1\}\}$. Notons que seules les classes d'équivalence des cellules identifiées avec au moins une autre cellule sont représentées.

2.3.2 Parallèle avec les prérequis d'implémentation relevés dans l'analyse

Nous justifions les choix de structure de données de la sous-section 2.3.1 par rapport aux prérequis d'implémentation relevés dans les paragraphes 2.1.2-c, 2.1.3-c et 2.1.4-c. Nous distinguons chaque catégorie de prérequis : structure de données, construction de matrices, modification de matrices, produit impliquant une matrice, fusion de matrices.

2.3.2-a Structure de données

Les deux seuls prérequis de cette catégorie concernent la représentation implicite de certaines valeurs des matrices. Pour rappel, ils sont :

— Structure-A, *Représentation en matrice creuse : les valeurs nulles sont implicites* ;
— Structure-B, *Représentation implicite de l'identité :* $\phi[x, x] = 1$ *est implicite*.

Matrice creuse. Considérons le prérequis Structure-A. Les coefficients sont représentés sous la forme de triplet (i, j, v) reliés entre eux par deux chaînages doubles. Pour représenter implicitement les valeurs nulles d'une matrice, il suffit de ne pas créer de maillons tels que $v = 0$. Ce prérequis est donc satisfait.

Identité implicite. Considérons le prérequis Structure-B. Les matrices sont munies d'un booléen *isIdentityVariation* indiquant si elles sont étudiées comme variation par rapport à l'identité. En particulier, pour une matrice M donnée, les coefficients de la forme $M[i, i] = 1$ sont ajoutés à l'image ou aux antécédents de n'importe quelle cellule d'indice i à condition qu'elle soit vivante dans le domaine et dans le codomaine de M. Ce prérequis est donc satisfait.

2.3.2-b Construction de matrices

Pour rappel, les prérequis de cette catégorie sont :

— Construction-A, *Construire une matrice* ;
— Construction-B, *Construction d'une matrice qui ne dépend pas de son nombre de colonnes* ;
— Construction-C, *Construction d'une matrice qui ne dépend pas de son nombre de lignes ni de son nombre de colonnes* ;
— Construction-D, *Construction d'une matrice qui ne dépend pas de son nombre de lignes* ;

Construction quelconque. Considérons le prérequis Construction-A. Une construction "quelconque" est une construction sans contraintes. Pour satisfaire ce prérequis, proposer une méthode de construction de matrices est donc suffisant. Par ailleurs, notons que pour construire une matrice, on fournit deux répertoires de cellules représentant son domaine et son codomaine, et l'on indique une valeur pour *isIdentityVariation*.

Construction sans dépendre de l'une ou l'autre des dimensions. Considérons les prérequis Construction-B et Construction-D. La contrainte est la même dans les deux cas : construire une matrice sans dépendre de l'une de ses dimensions. Ce prérequis est satisfait par les matrices temporaires car leurs points d'entrée sont stockés dans des `dictionnaires` contenant autant d'éléments que la matrice ne compte de lignes ou de colonnes **non nulles**. À sa construction, une matrice n'a aucune ligne ou colonne non nulle. Enfin, observons que toutes les matrices concernées par les prérequis Construction-B et Construction-D sont des matrices temporaires. Pour rappel, il s'agit de :

— i et r dans le calcul d'une SECE (cf. paragraphe 2.1.2-c). Ces deux matrices sont des matrices temporaires ;
— i^B, i^S et $f^{S,t}$ dans l'application du théorème SECE à l'identification (cf. paragraphe 2.1.3-c). i^B et i^S sont des matrices temporaires. La matrice $f^{S,t}$ est une matrice maintenue. Elle est concernée par ces prérequis car elle implique une fusion de matrices telle que l'une des matrices est $h^S i^B f^{S,t}$. Cette dernière est une matrice temporaire et c'est à elle que s'appliquent les prérequis de construction ;
— χ, χ^B, χ^S, $f^{S,t}$ et $g^{S,t}$ dans l'application du théorème SECE à la désidentification (cf. paragraphe 2.1.4-c). Les matrices χ, χ^B et χ^S sont temporaires. Les matrices $f^{S,t}$ et $g^{S,t}$ sont maintenues. Elles sont concernées par les prérequis de construction car elles impliquent chacune une fusion de matrice composée respectivement de $-h^{S,t}\chi^B f^S$ et $-g^{S,t}\chi^B h^S$, qui sont deux matrices temporaires.

Toutes les matrices concernées par les prérequis Construction-B et Construction-D sont temporaires. Ces prérequis sont satisfaits par les matrices temporaires. Le prérequis est donc satisfait.

Construction sans dépendre d'aucune des dimensions. Considérons le prérequis Construction-C. Les matrices qu'il concerne sont j et s, qui apparaissent dans le calcul d'une SECE (cf. paragraphe 2.1.2-c). Ces deux matrices sont temporaires, mais leur construction est problématique du fait de leur domaine et de leur codomaine : l'un et l'autre sont représentés par le même répertoire de cellule, mais avant et après une étape de construction. Dit autrement, il n'est pas possible d'avoir dans le même temps les deux états de ce répertoire de cellules sans qu'il soit dupliqué, et donc sans avoir une construction qui dépend du nombre de lignes ou du nombre de colonnes des matrices. Pour autant, nous avons vu dans l'analyse que ces matrices sont utilisées :

— dans l'application du théorème SECE à l'identification, et plus particulièrement pour le calcul de,
  · g^{t+1}, où s est utilisée dans deux produits $-s\partial^t rg$ et sg^t. Or, nous avons vu qu'il est en réalité possible de se passer de s pour ces deux calculs, en utilisant une opération de suppression de lignes ;
  · f^{t+1}, où j est utilisée dans un produit $f^t j$. Or, nous avons vu qu'il est en réalité possible de se passer de j pour ce calcul, en utilisant des opérations de somme de colonnes et de suppression de colonnes ;
— dans l'application du théorème SECE à la désidentification, et plus particulièrement pour le calcul de ,
  · χ, où s est utilisée pour un produit $s\partial^{t+1} r$. Or, nous avons vu qu'il est en réalité possible de se passer de s pour ce calcul, en utilisant une opération de suppression de lignes ;
  · f^{t+1}, où s est utilisée pour un produit $f^t s$. Or, nous avons vu qu'il est en réalité possible de se passer de s pour ce calcul, en utilisant une opération d'ajout de colonnes vides ;
  · g^{t+1}, où j est utilisé pour un produit jg^t. Or, nous avons vu qu'il est en réalité possible de se passer de j pour ce calcul, en utilisant une opération de duplication de lignes.

Au bout du compte, on observe que pour tous les calculs dans lesquels j et s apparaissent, il est possible de se passer des matrices en utilisant des opérations de modifications. Il n'est donc pas nécessaire de construire ces matrices et le prérequis Construction-C peut être ignoré du fait des choix d'implémentation que nous avons faits.

2.3.2-c Modification de matrices

Pour rappel, les prérequis de cette catégorie sont :

— Modification-A, *Suppression d'une ligne d'une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes* ;

— Modification-B, *Somme de deux colonnes dans une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes* ;

— Modification-C, *Suppression d'une colonne d'une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes* ;

— Modification-D, *Ajout de lignes vides dans une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes* ;

— Modification-E, *Ajout de colonnes vides dans une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes* ;

— Modification-F, *Duplication d'une ligne d'une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes* ;

— Modification-G, *Duplication d'une colonne d'une matrice sans dépendre de son nombre de lignes ni de son nombre de colonnes* ;

— Modification-H, *Suppression de coefficients dans une matrice sans dépendre de son nombre de lignes ou de son nombre de colonnes* ;

Du fait des structures de données que nous avons choisies pour la représentation matricielle, nous pouvons considérer que les opérations de modification de ligne et de colonne sont "symétriques", c'est-à-dire que si un prérequis est satisfait pour une certaine opération sur les lignes, alors il est satisfait pour la même opération sur les colonnes. En particulier, cela découle du fait que les points d'entrée en ligne et en colonne sont représentés de la même façon.

Suppression de lignes ou colonnes. Considérons le prérequis Modification-A. La suppression d'une ligne d'une matrice est une opération qui fait suite à la mort d'une cellule. Elle est utilisée sur g^t et ∂^t pour le calcul de g^{t+1} et ∂^{t+1}, dans l'application du théorème SECE à l'identification. Ces deux matrices sont des matrices maintenues. Soit i l'indice de la cellule supprimée dans sa base. L'opération consiste à :

— accéder au premier maillon de la ligne i au travers du deque contenant les points d'entrée de la matrice considérée. L'accès est donc en temps constant ;

— parcourir la ligne i maillon par maillon. Pour chaque maillon (i, j, v),

  · mettre à jour son voisinage de manière cohérente, c'est-à-dire que le voisin du bas de `top` devient `bottom` etc...

  · supprimer (i, j, v) et passer au maillon suivant.

Observons que le nombre de lignes de la matrice dans laquelle on fait la suppression n'influe pas sur cette opération. Le prérequis Modification-A est donc satisfait.

Remarque. Le prérequis Modification-C est satisfait par symétrie.

Somme de colonnes. Considérons le prérequis Modification-B. Les matrices concernées par ce prérequis sont ∂^t et f^t dans l'application du théorème SECE à l'identification (cf. paragraphe 2.1.3-c). Ces deux matrices sont des matrices maintenues. Soit M la matrice dans laquelle on somme une colonne d'indice j' dans une colonne d'indice j. L'opération consiste à :

— accéder aux premiers maillons des colonnes j et j' au travers d'un `deque`, donc en temps constant ;

— parcourir les colonnes j et j' maillon par maillon. Pour chaque maillon, on effectue une comparaison sur leur indice de ligne, en distinguant trois cas :

· il existe un maillon (i, j, v) et un maillon (i, j', v'). Dans ce cas, il suffit de changer la valeur de (i, j, v) en $(i, j, v + v')$ si $v + v' \neq 0$, sinon, le supprimer en reconnectant son voisinage de manière cohérente (le voisin du bas de `top` devient `bottom` etc. . .) ;

· il n'existe pas de maillon (i, j', v'). Dans ce cas il n'y a rien à faire ;

· il n'existe pas de maillon (i, j, v). Dans ce cas, il faut créer un maillon (i, j, v') et l'insérer dans la colonne j en le connectant à son voisinage.

On observe que seuls les nombres de valeurs non nulles des colonnes j et j' influent sur l'opération. Elle ne dépend ni du nombre de lignes, ni du nombre de colonnes de la matrice dans laquelle on somme. Le prérequis Modification-B est donc satisfait.

Remarque. Il n'y a pas de prérequis concernant la somme de lignes.

Ajout de lignes ou colonnes vides. Considérons le prérequis Modification-D. Les matrices concernées par ce prérequis sont :

— f^t dans l'application du théorème SECE à l'identification ;

— h^t dans l'application du théorème SECE à la désidentification.

Ces deux matrices sont des matrices maintenues. L'ajout d'une ligne vide est une opération qui fait suite à la création d'une cellule. Cette opération consiste à :

— créer la nouvelle cellule dans la base concernée. Un indice lui est attribué. Par convention, cet indice est la taille de la base ;

— ajouter un nouvel élément à la fin [11] des `deque` contenant les points d'entrée des matrices dans laquelle la base est utilisée. L'insertion en fin dans un `deque` est une opération en temps constant.

On constate que ni le nombre de lignes, ni le nombre de colonnes de la matrice dans laquelle on ajoute une ligne vide n'influent sur l'opération. Le prérequis Modification-D est donc satisfait.

Remarque. Le prérequis Modification-E est satisfait par symétrie.

Duplication d'une ligne ou d'une colonne. Considérons le prérequis Modification-F. La duplication d'une ligne est une opération qui consiste à créer une ligne vide i', puis à copier les maillons d'une ligne i dans la ligne i'. Les matrices concernées par ce prérequis sont ∂^t et ^{t} dans l'application du théorème SECE à la désidentification. Ces deux matrices sont des matrices maintenues. L'ajout d'une ligne vide dans une matrice maintenue étant traité dans le paragraphe précédent, nous nous intéressons uniquement à la copie des maillons. L'opération se fait en :

— accédant au premier maillon de la ligne i au travers des `deque` contenant les points d'entrée ;

11. L'ajout est toujours en fin car nous faisons en sorte que l'indice attribué aux nouvelles cellules soit toujours la taille de la base dans laquelle elles sont créées. Une autre solution consiste à attribuer, parfois, des indices négatifs à ces cellules, auquel cas l'ajout peut être au début. Les deux solutions sont couvertes par l'implémentation que nous donnons car, dans un `deque`, l'insertion en début est en temps constant. Cela est discuté plus en détail dans le paragraphe 2.3.2-e.

— parcourant la ligne i maillon par maillon. Pour chaque maillon (i, j, v) ainsi parcouru, créer le maillon (i', j, v) en le connectant à son voisinage.

On constate que ni le nombre de lignes, ni le nombre de colonnes de la matrice dans laquelle on duplique une ligne n'influent sur l'opération. Le prérequis Modification-F est donc satisfait.

Remarque. Le prérequis Modification-G est satisfait par symétrie.

Suppression de coefficients. Considérons le prérequis Modification-H. La matrice concernée par ce prérequis est ∂^t dans l'application du théorème SECE à la désidentification. Soit (i, j, v) le maillon représentant le coefficient que l'on souhaite supprimer. L'opération consiste à :

— accéder au premier maillon de la ligne i ;

— parcourir la ligne i maillon par maillon jusqu'à trouver (i, j, v) ;

— mettre à jour le voisinage de (i, j, v) de manière cohérente. Par exemple, le voisin du bas de `top` devient `bottom` etc. . .

— supprimer le maillon (i, j, v).

On constate que ni le nombre de lignes, ni le nombre de colonnes de la matrice dans laquelle on duplique une ligne n'influent sur l'opération. Le prérequis Modification-H est donc satisfait.

Remarque. La suppression peut se faire en parcourant la colonne j. L'algorithme est identique.

2.3.2-d Produit impliquant une matrice

Pour rappel, les prérequis de cette catégorie sont :

— Produit-A, *Produit matriciel $A \times B$ en parcourant les lignes de A* ;

— Produit-B, *Produit matriciel $A \times B$ en parcourant les colonnes de B* ;

— Produit-C, *Produit d'une matrice avec un scalaire* ;

— Produit-D, *Produit d'une matrice avec un scalaire, sans dépendre de son nombre de lignes* ;

— Produit-E, *Produit d'une matrice avec un scalaire, sans dépendre de son nombre de colonnes* ;

Produit en ne parcourant qu'une dimension. Considérons le prérequis Produit-A. Ce prérequis concerne :

— éventuellement la matrice ∂ dans le calcul d'une SECE (cf. paragraphe 2.1.2-c) ;

— les matrices i^B, i^S, $f^{S,t}$, $g^{S,t}$ et $h^{S,t}$ dans l'application du théorème SECE à l'identification (cf. paragraphe 2.1.3-c).

Dans tous ces cas, les produits $A \times B$ impliquent une matrice A temporaire et une matrice B maintenue. De plus, le résultat de ce produit est systématiquement une matrice C temporaire fusionnée avec une autre matrice. L'opération consiste à :

— créer une matrice C initialement vide, et qui représente le résultat du produit. Cette matrice est temporaire, sa construction est en temps constant ;

— accéder au premier maillon de chaque ligne de A en parcourant le `dictionnaire` contenant les points d'entrée en ligne de A. Soit (i, j, v) un tel maillon,

· accéder au premier maillon de la ligne j de B à partir du deque contenant les points d'entrée (donc avec une complexité en temps constant) et parcourir cette ligne maillon par maillon. Soit (j, k, v') un tel maillon ;

▷ S'il existe déjà un maillon (i, k, v'') dans C, le modifier pour $(i, k, v'' + v * v')$;

▷ Sinon, créer un maillon $(i, k, v * v')$ et l'écrire dans C.

Observons que les paramètres qui influent sur la complexité en temps de ce produit sont le nombre de lignes de A et les plus grands nombres de valeurs explicites dans une ligne de A et dans une ligne de B. Le prérequis Produit-A est donc satisfait.

Remarque. Le prérequis Produit-B est satisfait par symétrie. Plus précisément, les matrices concernées par le prérequis Produit-B sont :

— éventuellement ∂ dans le calcul d'une SECE (cf. paragraphe 2.1.2-c) ;

— g^t et h^t dans l'application du théorème SECE à l'identification (cf. paragraphe 2.1.3-c) ;

— χ, χ^B, χ^S, $g^{S,t}$, $f^{S,t}$ et $h^{S,t}$ dans l'application du théorème SECE à l'identification (cf. paragraphe 2.1.4-c) ;

Dans tous ces cas, les produits $A \times B$ impliquent une matrice A maintenue et une matrice B temporaire, d'où la symétrie.

Produit d'une matrice avec un scalaire sans dépendre d'une dimension. Considérons les prérequis Produit-C, Produit-D et Produit-E. Le produit d'une matrice avec un scalaire α non nul consiste à :

— accéder aux premiers maillons d'une ligne au travers de la structure stockant ses points d'entrée ;

— parcourir la ligne maillon par maillon. Soit (i, j, v) un tel maillon. On le modifie en $(i, j, v * \alpha)$.

L'opération peut se réaliser avec un parcours de colonnes. On observe donc que seule l'une des dimensions influe sur cette opération, les prérequis Produit-C, Produit-D et Produit-E sont donc satisfaits.

2.3.2-e Fusion de matrices

Rappelons que toute fusion de matrice consiste à modifier une matrice existante (cf. paragraphe 2.1.1-e). Les prérequis de cette catégorie sont :

— Fusion-A, *Fusion de matrices* $D = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$;

— Fusion-B, *Fusion de matrices* $B = \begin{bmatrix} A & B \end{bmatrix}$;

— Fusion-C, *Fusion de matrices* $A = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$. ;

— Fusion-D, *Fusion de matrices* $A = \begin{bmatrix} A & B \end{bmatrix}$;

— Fusion-E, *Fusion de matrices* $A = \begin{bmatrix} A \\ B \end{bmatrix}$;

Dans tous les cas rencontrés dans l'analyse, les matrices fusionnées sont temporaires et la matrice modifiée est maintenue. Cette opération se décompose en deux étapes :

1. la mise à jour des répertoires de cellules utilisées par la matrice modifiée ;
2. la "connexion" des maillons des *matrices fusionnées* à la *matrice modifiée*.

Soient deux bases de cellules $B_A = \{\sigma_{0_A}, \ldots, \sigma_{q_A}\}$ et $B_D = \{\sigma_{0_D}, \ldots, \sigma_{q_D}\}$, et les matrices :

— $A : B_A \to B_A$;
— $B : B_A \to B_D$;
— $C : B_D \to B_A$;
— $D : B_D \to B_D$.

Considérons le prérequis Fusion-A, c'est-à-dire que A, B et C sont des matrices temporaires, et D est une matrice maintenue. Avant la fusion, les répertoires de cellules de B_A et B_D sont :

B_A	D_A	L_A
σ_{0_A}	$(vrai, -, 1)$	A
$\vdots$	$\vdots$	B
σ_{q_A}	$(vrai, q_A - 1, -)$	C
`first`	0	

B_D	D_D	L_D
σ_{0_D}	$(vrai, -, 1)$	B
$\vdots$	$\vdots$	C
σ_{q_D}	$(vrai, q_D - 1, -)$	D
`first`	0	

Mise à jour des répertoires de cellules. L'étape 1 de la fusion consiste à ajouter les cellules de B_A dans B_D, c'est-à-dire que $B_D = B_A \oplus B_D = \{\sigma_{0_D}, \ldots, \sigma_D, \sigma_{0_A}, \ldots, \sigma_{q_A}\}$. Du côté des répertoires de cellules, la modification de B_D revient à :

— ajouter q_A éléments dans D_D. L'indice de chaque nouvel élément (et donc de chaque nouvelle cellule) est la taille de D_D avant son insertion ;
— mettre à jour les matrices de L_D en conséquence de manière cohérente. Plus précisément, cela veut dire qu'on ajoute q_A points d'entrée dans les matrices maintenues, D en l'occurrence.

Après la mise à jour de B_D, les matrices sont telles que :

— $A : B_A \to B_A$;
— $B : B_A \to B_A \oplus B_D$;
— $C : B_A \oplus B_D \to B_A$;
— $D : B_A \oplus B_D \to B_A \oplus B_D$.

et les répertoires de cellules sont tels que :

B_A	D_A	L_A
σ_{0_A}	$(vrai, -, 1)$	A
$\vdots$	$\vdots$	B
σ_{q_A}	$(vrai, q_A - 1, -)$	C
`first`	0	

$B_D = B_A \oplus B_D$	D_D	L_D
σ_{0_D}	$(vrai, -, 1)$	B
$\vdots$	$\vdots$	C
σ_{q_D}	$(vrai, q_D - 1, q_D + 1)$	D
σ_{0_A}	$(vrai, q_D, q_D + 2)$	
$\vdots$	$\vdots$	
σ_{q_A}	$(vrai, q_A - 1, -)$	
`first`	0	

À l'issue de l'étape 1, le répertoire de cellules représentant B_D est modifié pour toutes les matrices l'utilisant (c'est-à-dire celles de la liste L_D de B_D). Plus précisément :

— aucun point d'entrée n'est ajouté aux matrices temporaires, donc à A, B et C ;

— q_A points d'entrée sont ajoutés aux matrices maintenues, donc D. Ils pointent sur le vide.

Remarque. Il est important de noter que le fait d'attribuer comme indice aux nouvelles cellules insérées dans une base la taille de cette base avant leur insertion est un choix qui a pour conséquence que les fusions de matrices $A = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$ et $D = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$ sont traitées de manière similaire. Plus précisément, elles consistent toutes deux à :

— insérer les nouvelles cellules en fin du deque du répertoire de cellules modifié ;

— insérer les nouveaux points d'entrée en fin de deque dans les matrices concernées.

En conséquence, les lignes/colonnes d'une matrice induites par la création d'une cellule sont toujours insérées en bas ou à droite, ce qui ne respecte pas la notation $D = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$. Pour respecter la notation, on peut autoriser l'attribution d'indices négatifs aux cellules créées. Dans ce cas, il est possible d'insérer des lignes/colonnes en haut/à gauche ou en bas/à droite en évitant d'avoir à ré-indexer les cellules déjà existantes. Les deux solutions sont couvertes par l'implémentation que nous donnons, et passer de l'une à l'autre n'implique aucune modification des structures de données utilisées pour la représentation matricielle ou les répertoires de cellules. En effet, un deque est non seulement muni d'une opération d'insertion en fin en temps constant, mais aussi d'une insertion en début en temps constant. Notons cependant que le chaînage double des maillons n'est nécessaire que dans le cas où l'on utilise des indices négatifs. En résumé, il est important de faire correspondre à chaque cellule un indice unique dans une base, et que toutes les matrices utilisant cette base respectent ces indices, mais la valeur de l'indice n'a pas d'importance.

Connexion des maillons. Au cours de l'étape 2, les maillons des matrices A, B et C sont dupliqués et connectés à ceux de la matrice D. Rappelons qu'à ce stade, les points d'entrée dans D correspondant aux nouvelles cellules de B_D sont déjà créés et pointent sur le vide. L'opération consiste alors à :

— parcourir les lignes de la matrice B, ligne par ligne puis maillon par maillon. Pour chaque maillon (i, j, v) ainsi parcouru, le dupliquer et remplacer i par i', où i' correspond à l'indice de la i-ème cellule de B_A dans $B_A \oplus B_D$. Ensuite, connecter (i', j, v) à ses voisins dans D. ;

— parcourir les colonnes de la matrice C, colonne par colonne puis maillon par maillon. Pour chaque maillon (i, j, v) ainsi parcouru, le dupliquer et remplacer j par j', où j' correspond à l'indice de la j-ème cellule de B_A dans $B_A \oplus B_D$. Ensuite, connecter (i, j', v) à ses voisins dans D. ;

— parcourir les lignes ou colonnes de la matrice A, vecteur par vecteur puis maillon par maillon. Pour chaque maillon (i, j, v) ainsi parcouru, le dupliquer et remplacer i par i' et j par j', où i' et j' correspondent, respectivement, aux indices de la i-ème cellule et de la j-ème cellule de B_A dans $B_A \oplus B_D$. Ensuite, connecter (i', j', v) à ses voisins dans D.

Observons qu'à aucun moment dans cet algorithme ne sont parcourus :

— les lignes ou colonnes de D ;

— les maillons de D. Les seuls maillons de D auxquels on accède se situent aux extrémités de la matrice, et sont accessibles en temps constant via les points d'entrée stockés dans les deque de D.

En conséquence, le prérequis Fusion-A est satisfait. Les autres prérequis de fusion sont satisfaits car ils consistent à appliquer le même algorithme, en mettant à jour des bases différentes :

— la Fusion-B, *Fusion de matrices* $B = \begin{bmatrix} A & B \end{bmatrix}$, est équivalente à, par exemple, la fusion $B = \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix}$. Elle se réalise avec le même algorithme mais en modifiant les bases de B ;

— la Fusion-C, *Fusion de matrices* $A = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$., est équivalente à $D = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$, en modifiant les bases de A ;

— la Fusion-D, *Fusion de matrices* $A = \begin{bmatrix} A & B \end{bmatrix}$, est équivalente à $A = \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix}$, en modifiant les bases de A ; la Fusion-E, *Fusion de matrices* $A = \begin{bmatrix} A \\ B \end{bmatrix}$, est équivalente à $A = \begin{bmatrix} A & 0 \\ C & 0 \end{bmatrix}$, en modifiant les bases de A.

Remarque. Notons que l'étape 2 est dépendante de l'étape 1 : avant de connecter les maillons des matrices fusionnées, il faut au préalable calculer $B_A \oplus B_D$. Dans le théorème SECE, certaines fusions de matrices impliquent les mêmes modifications de bases pour des matrices différentes (par exemple, les fusions de $f^{S,t+1}$ et g^{t+1} impliquent toutes deux de modifier B_{C^t} de la même façon, $B_{C^{t+1}} = B_C \oplus B_{C^t}$). Dans ce cas, l'étape 1 est effectuée une unique fois, en amont de toute étape 2.

2.3.3 Exemple d'utilisation des structures de données

Cette sous-section illustre sur un exemple les répertoires de cellules et la représentation matricielle que nous avons conçus. L'exemple choisi est une itération de l'application du théorème SECE à l'identification, et est illustré sur la figure 2.22. L'identification est caractérisée par :

$$I^0 = \{\{v_0\}, \{v_1\}, \{v_3, v_2\}\}$$
$$I^1 = \{\{e_0\}, \{e_2\}, \{e_3\}, \{e_4\}, \{e_5, e_1\}\}$$
$$I^2 = \{\{f_0\}, \{f_1\}\}$$

et $\zeta : B_{C^t} \to B_{C^t}$ telle que :

— $v_0\zeta = v_0$, $v_1\zeta = v_1$, $v_2\zeta = v_3\zeta = v_3$;

— $e_0\zeta = e_0$, $e_1\zeta = e_5\zeta = e_5$, $e_2\zeta = e_2$, $e_3\zeta = e_3$, $e_4\zeta = e_4$;

— $f_0\zeta = f_0$, $f_1\zeta = f_1$.

Remarque. Pour simplifier cette sous-section, nous ne distinguons pas les dimensions des cellules. En particulier, les cellules sont indicées de manière continue. Par exemple, v_3 est indicé 3 et e_1 est indicé 4. Par ailleurs, cet exemple est fourni avec le logiciel que nous avons développé (présenté dans la sous-section suivante 2.3.4), dans les fichiers `figure3_20.hominc` et `figure3_20.cprocess`.

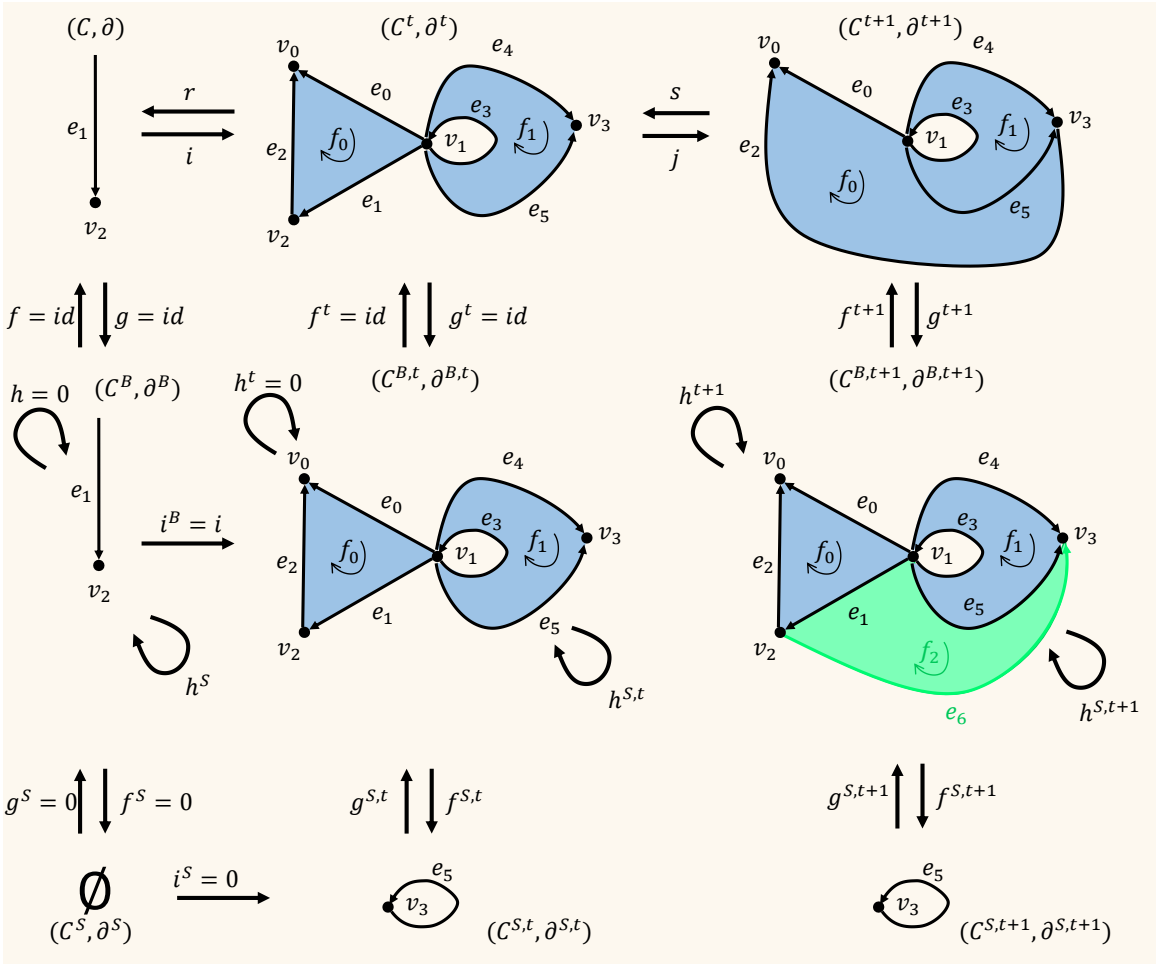


FIGURE 2.22 – Application du théorème SECE dans le cas de l'identification. En vert, les générateurs de $(C^{B,t+1}, \partial^{B,t+1})$ en plus par rapport à $(C^{B,t}, \partial^{B,t})$. Le passage de t à $t+1$ se fait par identification de v_3 avec v_2 et de e_5 avec e_1. Les cellules v_3 et e_5 sont survivantes.

2.3.3-a Données de départ.

Les données de départ de l'application du théorème SECE sont l'équivalence homologique

$$\gamma^t : (C^t, \partial^t) \overset{\rho^t}{\Longleftarrow} (C^{B,t}, \partial^{B,t}) \overset{\rho^{S,t}}{\Longrightarrow} (C^{S,t}, \partial^{S,t})$$

et l'opération d'identification caractérisée par I et $\zeta : B_{C^t} \to B_{C^t}$. La représentation de l'identification est illustrée sur la figure 2.23.

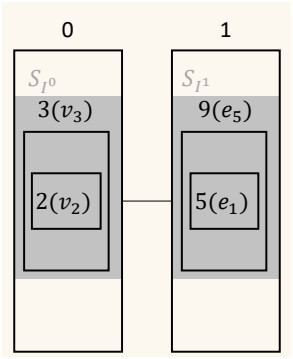


FIGURE 2.23 – L'identification caractérisée par I et $\zeta : B_{C^t} \to B_{C^t}$ représentée par un `dictionnaire` S_I dont les clefs sont 0 et 1 et dont les valeurs sont, respectivement, S_{I^0} et S_{I^1}. S_{I^0} est un `dictionnaire` dont la clef est 3 et la valeur est un `tableau` de taille 1 contenant 2. S_{I^1} est un `dictionnaire` dont la clef est 9 et la valeur est un `tableau` de taille 1 contenant 5. Les deux dictionnaires S_{I^0} et S_{I^1} sont mis en évidence par un fond gris.

Les répertoires de cellules des bases B_{C^t}, $B_{C^{B,t}}$ et $B_{C^{S,t}}$ sont :

B_{C^t}	$D_{B_{C^t}}$	$L_{B_{C^t}}$
v_0	$(vrai, -, 1)$	∂^t
v_1	$(vrai, 0, 2)$	f^t
v_2	$(vrai, 1, 3)$	g^t
v_3	$(vrai, 2, 4)$	
e_0	$(vrai, 3, 5)$	
e_1	$(vrai, 4, 6)$	
e_2	$(vrai, 5, 7)$	
e_3	$(vrai, 6, 8)$	
e_4	$(vrai, 7, 9)$	
e_5	$(vrai, 8, 10)$	
f_0	$(vrai, 9, 11)$	
f_1	$(vrai, 10, -)$	
`first`	0	

$B_{C^{B,t}}$	$D_{B_{C^{B,t}}}$	$L_{B_{C^{B,t}}}$
v_0	$(vrai, -, 1)$	$\partial^{B,t}$
v_1	$(vrai, 0, 2)$	f^t
v_2	$(vrai, 1, 3)$	g^t
v_3	$(vrai, 2, 4)$	h^t
e_0	$(vrai, 3, 5)$	$h^{S,t}$
e_1	$(vrai, 4, 6)$	$f^{S,t}$
e_2	$(vrai, 5, 7)$	$g^{S,t}$
e_3	$(vrai, 6, 8)$	
e_4	$(vrai, 7, 9)$	
e_5	$(vrai, 8, 10)$	
f_0	$(vrai, 9, 11)$	
f_1	$(vrai, 10, -)$	
`first`	0	

$B_{C^{S,t}}$	$D_{B_{C^{S,t}}}$	$L_{B_{C^{S,t}}}$
v_0	$(faux, -, -)$	$\partial^{S,t}$
v_1	$(faux, -, -)$	$f^{S,t}$
v_2	$(faux, -, -)$	$g^{S,t}$
v_3	$(vrai, -, 9)$	$h^{S,t}$
e_0	$(faux, -, -)$	
e_1	$(faux, -, -)$	
e_2	$(faux, -, -)$	
e_3	$(faux, -, -)$	
e_4	$(faux, -, -)$	
e_5	$(vrai, 3, -)$	
f_0	$(faux, -, -)$	
f_1	$(faux, -, -)$	
`first`	3	

On suppose que l'on souhaite calculer les générateurs d'homologie, toutes les matrices de γ^t doivent donc être maintenues. Les matrices de γ^t sont : ∂^t, f^t, g^t, h^t, $\partial^{B,t}$, $f^{S,t}$, $g^{S,t}$, $h^{S,t}$ et $\partial^{S,t}$. En particulier, f^t et g^t sont des matrices identité et h^t est une matrice nulle. Les autres matrices sont données ci-dessous.

$$\partial^t = \partial^{B,t}$$

	v_0	v_1	v_2	v_3	e_0	e_1	e_2	e_3	e_4	e_5	f_0	f_1
v_0	.	.	.	.	.	.	.	.	.	.	.	.
v_1	.	.	.	.	.	.	.	.	.	.	.	.
v_2	.	.	.	.	.	.	.	.	.	.	.	.
v_3	.	.	.	.	.	.	.	.	.	.	.	.
e_0	1	−1	.	.	.	.	.	.	.	.	.	.
e_1	.	−1	1	.	.	.	.	.	.	.	.	.
e_2	1	.	−1	.	.	.	.	.	.	.	.	.
e_3	.	.	.	.	.	.	.	.	.	.	.	.
e_4	.	−1	.	1	.	.	.	.	.	.	.	.
e_5	.	−1	.	1	.	.	.	.	.	.	.	.
f_0	.	.	.	.	−1	1	1	.	.	.	.	.
f_1	.	.	.	.	.	.	.	1	1	−1	.	.

$$f^{S,t}$$

	v_3	e_5
v_0	1	.
v_1	1	.
v_2	1	.
v_3	1	.
e_0	.	.
e_1	.	.
e_2	.	.
e_3	.	1
e_4	.	.
e_5	.	1
f_0	.	.
f_1	.	.

$$g^{S,t}$$

	v_0	v_1	v_2	v_3	e_0	e_1	e_2	e_3	e_4	e_5	f_0	f_1
v_3	.	.	.	1	.	.	.	.	.	.	.	.
e_5	.	.	.	.	.	.	.	.	−1	1	.	.

$$\partial^{S,t}$$

	v_3	e_5
v_3	.	.
e_5	.	.

$$
\begin{array}{c}
h^{S,t} \\
v_0 \\
v_1 \\
v_2 \\
v_3 \\
e_0 \\
e_1 \\
e_2 \\
e_3 \\
e_4 \\
e_5 \\
f_0 \\
f_1
\end{array}
\begin{array}{c}
\begin{array}{cccccccccccc}
v_0 & v_1 & v_2 & v_3 & e_0 & e_1 & e_2 & e_3 & e_4 & e_5 & f_0 & f_1
\end{array} \\
\left[\begin{array}{cccccccccccc}
. & . & . & . & . & 1 & 1 & . & -1 & . & . & . \\
. & . & . & . & . & . & . & . & -1 & . & . & . \\
. & . & . & . & . & 1 & . & . & -1 & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & -1 & . \\
. & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & 1 \\
. & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & .
\end{array}\right]
\end{array}
$$

Les représentations de ces matrices avec les structures de données que nous avons choisies sont illustrées sur les figures 2.24, 2.25, 2.26, 2.27 et 2.28.

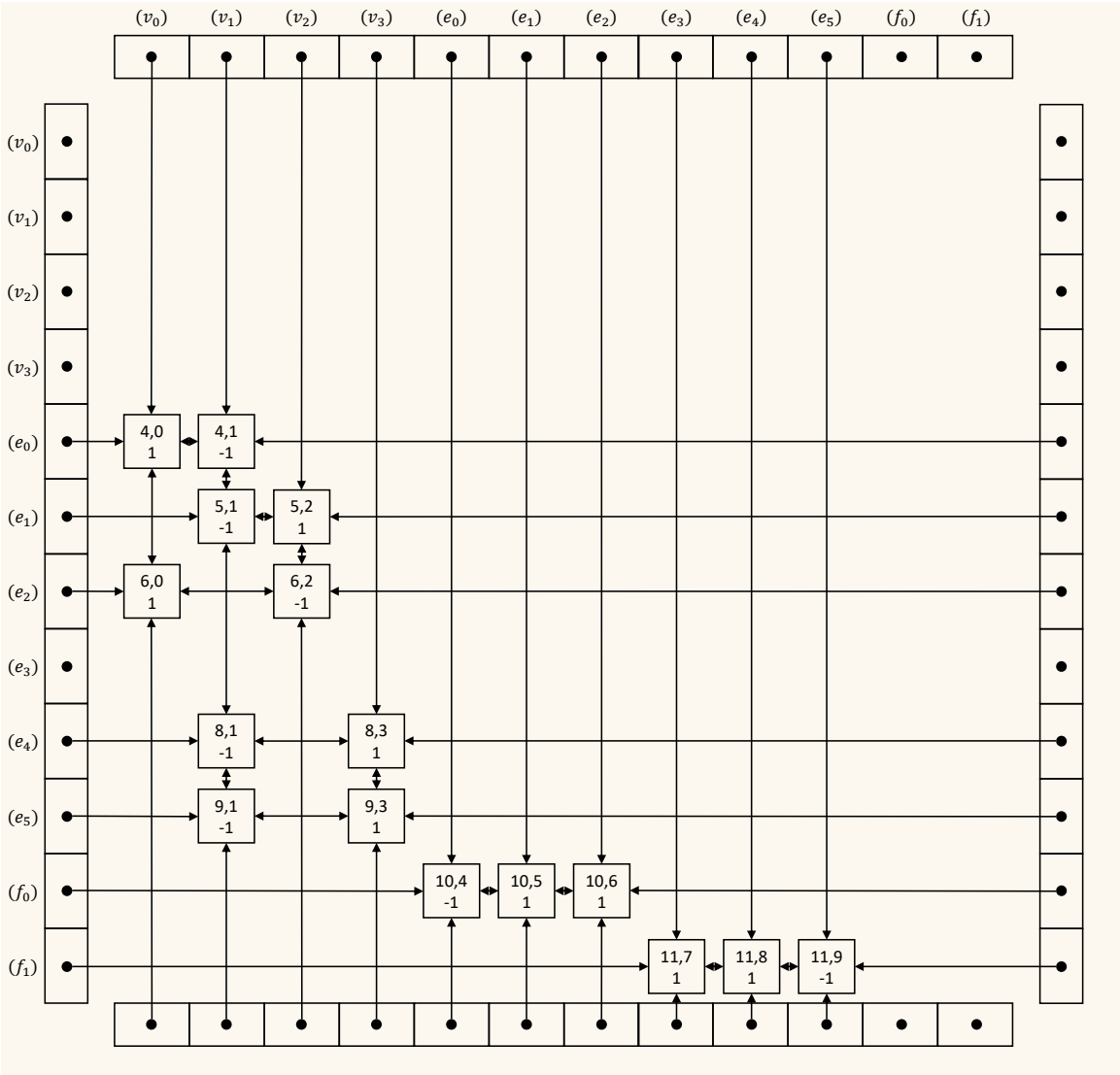


FIGURE 2.24 – La représentation de $\partial^t : B_{C^t} \to B_{C^t}$ (identique à celle de $\partial^{B,t} : B_{C^{B,t}} \to B_{C^{B,t}}$). Les labels à côté des deque contenant les points d'entrée sont là pour rappeler quelle cellule représente un indice de ligne ou de colonne.

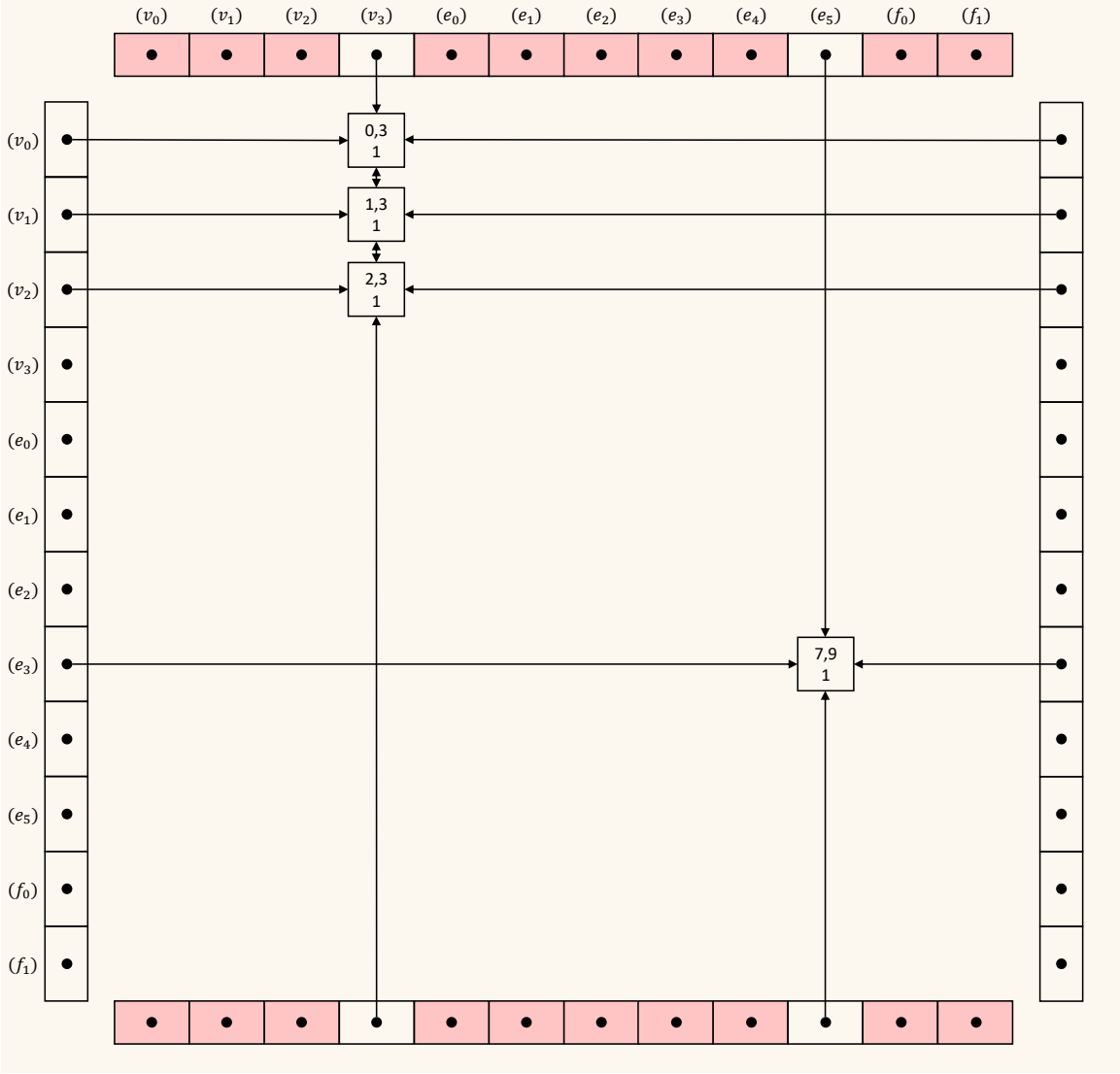


FIGURE 2.25 – La représentation de $f^{S,t}$: $B_{C^{B,t}} \rightarrow B_{C^{S,t}}$. Les labels à côté des deque contenant les points d'entrée sont là pour rappeler quelle cellule représente un indice de ligne ou de colonne. Les points d'entrée avec un fond rouge indiquent que la cellule qu'ils représentent est marquée comme morte dans le répertoire de cellules auquel elle appartient. Notons que $f^{S,t}[3,3] = 1$ et $f^{S,t}[9,9] = 1$ sont implicites car $isIdentityVariation$ vaut vrai pour $f^{S,t}$.

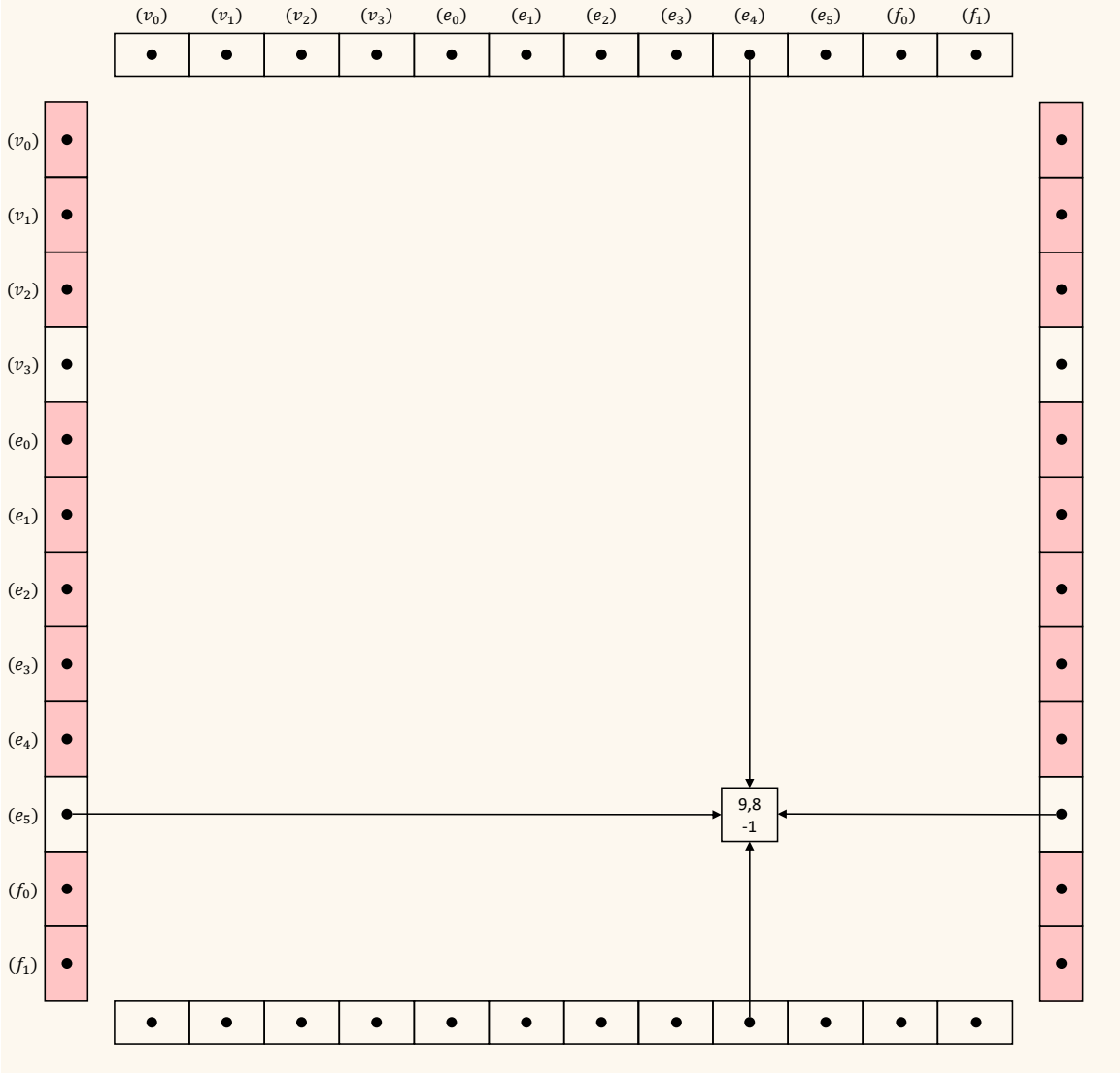


FIGURE 2.26 – La représentation de $g^{S,t}$: $B_{C^{S,t}} \rightarrow B_{C^{B,t}}$. Les labels à côté des deque contenant les points d'entrée sont là pour rappeler quelle cellule représente un indice de ligne ou de colonne. Les points d'entrée avec un fond rouge indiquent que la cellule qu'ils représentent est marquée comme morte dans le répertoire de cellules auquel elle appartient. Notons que $g^{S,t}[3,3] = 1$ et $g^{S,t}[9,9] = 1$ sont implicites car $isIdentityVariation$ vaut vrai pour $g^{S,t}$.

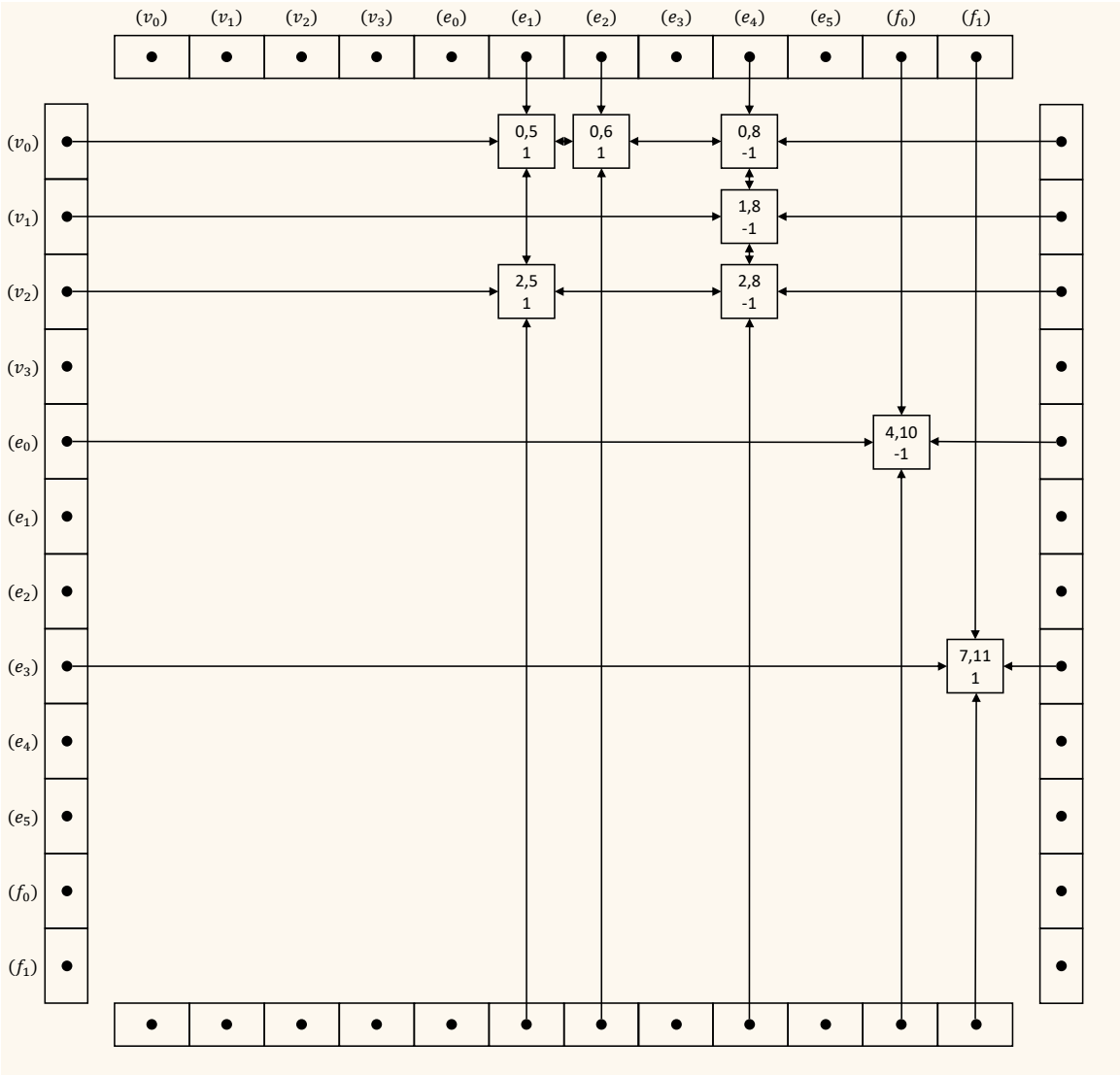


FIGURE 2.27 – La représentation de $h^{S,t} : B_{C^{B,t}} \to B_{C^{B,t}}$. Les labels à côté des deque contenant les points d'entrée sont là pour rappeler quelle cellule représente un indice de ligne ou de colonne.

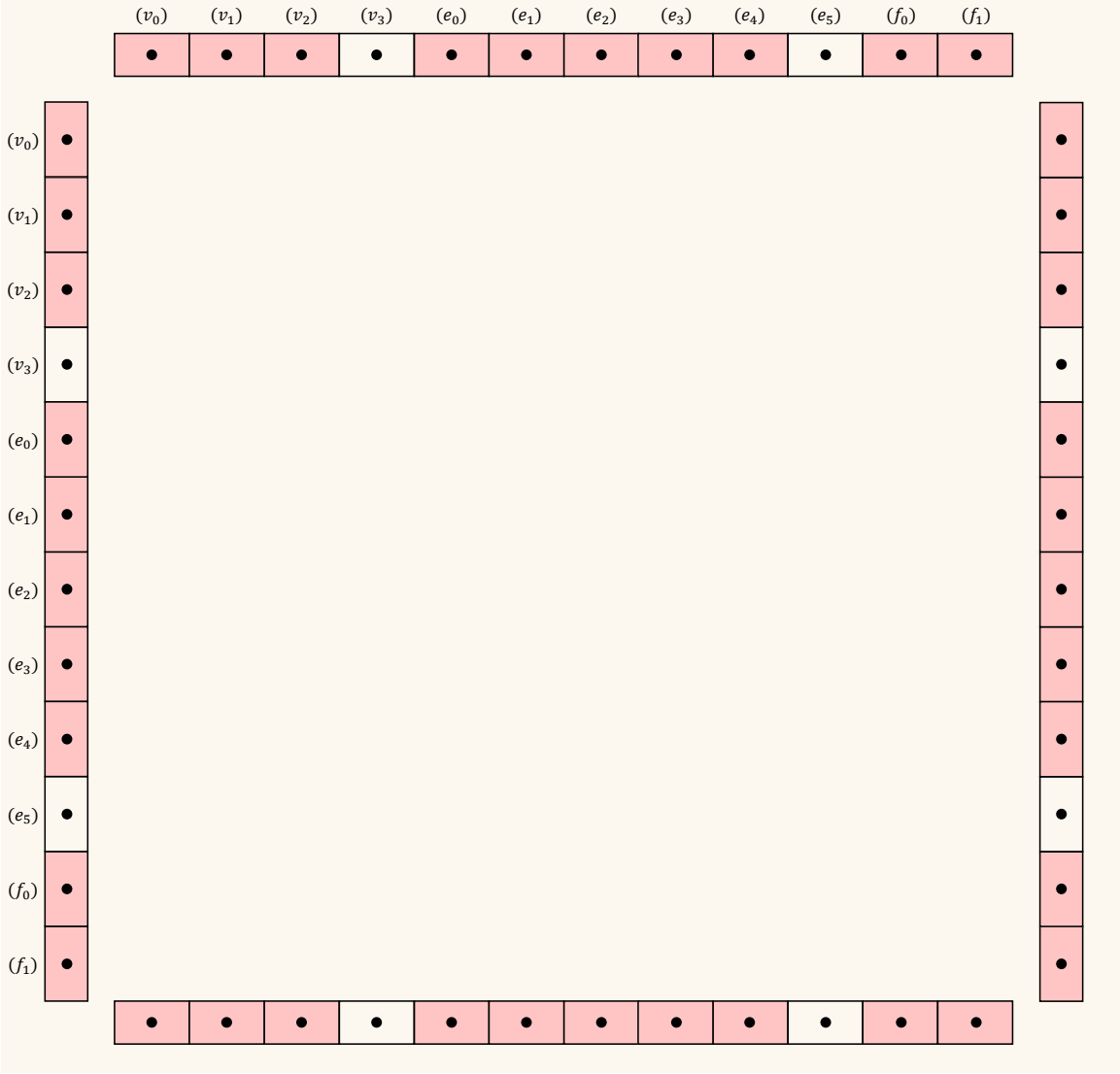


FIGURE 2.28 – La représentation de $\partial^{S,t} : B_{C^{S,t}} \to B_{C^{S,t}}$. Les labels à côté des deque contenant les points d'entrée sont là pour rappeler quelle cellule représente un indice de ligne ou de colonne. Les points d'entrée avec un fond rouge indiquent que la cellule qu'ils représentent est marquée comme morte dans le répertoire de cellules auquel elle appartient.

2.3.3-b Calcul de la SECE.

Les premières données calculées sont celles de la SECE

$$(C, \partial) \underset{\mathsf{r}}{\overset{\mathsf{i}}{\leftrightarrows}} (C^t, \partial^t) \underset{\mathsf{s}}{\overset{\mathsf{j}}{\leftrightarrows}} (C^{t+1}, \partial^{t+1})$$

et plus précisément, dans l'ordre :

— le répertoire de cellules représentant B_C ;

— les matrices temporaires i et r ;

— la matrice temporaire ∂ et l'équivalence homologique γ.

Calcul du répertoire de cellules représentant B_C. L'objectif est de calculer $B_C = \{v_2, e_1\}$. Ce calcul consiste dans un premier temps à construire un répertoire de cellules vide, c'est-à-dire composé d'un deque D_{B_C} vide et d'une `liste` L_{B_C} vide. Ensuite, on parcourt I au travers de S_I, et on ajoute autant d'éléments à D_{B_C} que S_I ne compte de cellules non survivantes. Le répertoire de cellules représentant B_C est alors :

B_C	D_{B_C}	L_{B_C}
v_2	$(vrai, -, 1)$	
e_1	$(vrai, 0, -)$	
`first`	0	

Calcul des matrices i et r. L'objectif est de calculer les matrices i et r suivantes :

$$i = \begin{array}{c} \\ v_2 \\ e_1 \end{array} \begin{array}{c} \begin{array}{cccccccccccc} v_0 & v_1 & v_2 & v_3 & e_0 & e_1 & e_2 & e_3 & e_4 & e_5 & f_0 & f_1 \end{array} \\ \begin{bmatrix} . & . & -1 & 1 & . & . & . & . & . & . & . & . \\ . & . & . & . & . & -1 & . & . & . & 1 & . & . \end{bmatrix} \end{array} \qquad r = \begin{array}{c} \\ v_0 \\ v_1 \\ v_2 \\ v_3 \\ e_0 \\ e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ f_0 \\ f_1 \end{array} \begin{array}{c} \begin{array}{cc} v_2 & e_1 \end{array} \\ \begin{bmatrix} . & . \\ . & . \\ -1 & . \\ . & . \\ . & . \\ . & -1 \\ . & . \\ . & . \\ . & . \\ . & . \\ . & . \\ . & . \end{bmatrix} \end{array}$$

Dans un premier temps, ces matrices sont construites, vides. Pour rappel, ce sont deux matrices temporaires. Étant donnés les répertoires de cellules de B_C et B_{C^t}, leur construction se résume à :

— construire des `dictionnaires` vides destinés à stocker les points d'entrée ;

— ajouter un pointeur sur i et r dans L_{B_C} et dans $L_{B_{C^t}}$.

À l'issue de ces constructions, on dispose des deux matrices temporaires i et r, vides, et des répertoires de cellules de B_C et B_{C^t} suivants :

B_C	D_{B_C}	L_{B_C}
v_2	$(vrai, -, 1)$	i
e_1	$(vrai, 0, -)$	r
`first`	0	

B_{C^t}	$D_{B_{C^t}}$	$L_{B_{C^t}}$
v_0	$(vrai, -, 1)$	∂^t
v_1	$(vrai, 0, 2)$	f^t
v_2	$(vrai, 1, 3)$	g^t
v_3	$(vrai, 2, 4)$	i
e_0	$(vrai, 3, 5)$	r
e_1	$(vrai, 4, 6)$	
e_2	$(vrai, 5, 7)$	
e_3	$(vrai, 6, 8)$	
e_4	$(vrai, 7, 9)$	
e_5	$(vrai, 8, 10)$	
f_0	$(vrai, 9, 11)$	
f_1	$(vrai, 10, -)$	
`first`	0	

Dans un second temps, on parcourt I au travers de S_I. Plus précisément :

— on initialise un entier $k = 0$. Cet entier sert à stocker l'indice d'une cellule non-survivante dans le parcours de S_I. Il représente[12] l'indice de la cellule dans B_C ;

— on parcourt S_I, c'est-à-dire que,

· l'on parcourt S_{I^0}. Il n'y a qu'une clef 3 contenant un tableau de taille 1, dont la valeur est 2. On en déduit les maillons à écrire :

▷ $(k = 0, 2, -1)$ et $(k = 0, 3, 1)$ dans i. Ils représentent $v_2 i = v_3 - v_2$;

▷ $(2, k = 0, -1)$ dans r. Il représente $v_2 r = -v_2$;

· l'on incrémente k à chaque cellule non-survivante parcourue. À ce stade, k est donc égal à 1 ;

· l'on parcourt S_{I^1}. Il n'y a qu'une clef 9 contenant un tableau de taille 1, dont la valeur est 5. On en déduit les maillons à écrire :

▷ $(k = 1, 5, -1)$ et $(k = 1, 9, 1)$ dans i. Ils représentent $e_1 i = e_5 - e_1$;

▷ $(5, k = 1, -1)$ dans r. Il représente $e_1 r = -e_1$.

Les représentations matricielles de i et r qu'il en résulte sont illustrées sur les figures 2.29 et 2.30.

12. Ceci n'est possible que parce que les parcours de S_I, S_{I^0} et S_{I^1} sont déterministes (cf. paragraphe 2.3.1-c), c'est-à-dire que l'on accède d'abord à v_2 puis à e_1.

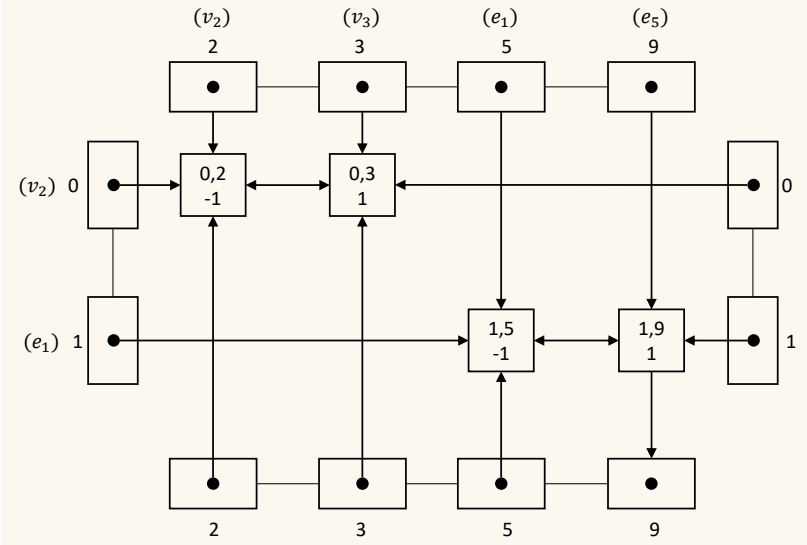


FIGURE 2.29 – La matrice temporaire $i : B_C \to B_{C^t}$. Elle représente les relations $v_2 i = v_3 - v_2$ et $e_1 i = e_5 - e_1$. Elle est identique à la matrice $i^B : B_{C^B} \to B_{C^B}$. Les labels des points d'entrée correspondent aux clefs des `dictionnaires`. Entre parenthèses, les cellules que représentent les lignes/colonnes.

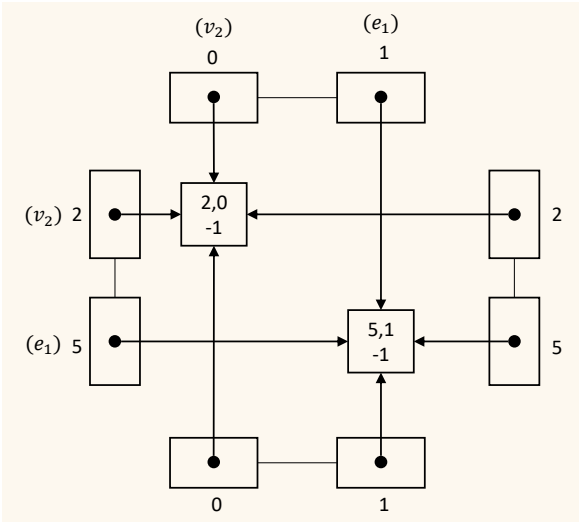


FIGURE 2.30 – La matrice temporaire $r : B_{C^t} \to B_C$. Elle représente les relations $v_2 r = -v_2$ et $e_1 r = -e_1$. Les labels des points d'entrée correspondent aux clefs des `dictionnaires`. Entre parenthèses, les cellules que représentent les lignes/colonnes.

Calcul de la matrice ∂ et de l'équivalence homologique γ. L'objectif est de calculer la matrice $\partial : B_C \to B_C$ suivante :

$$\begin{array}{c} \partial \\ v_2 \\ e_1 \end{array} \begin{array}{c} \begin{array}{cc} v_2 & e_1 \end{array} \\ \begin{bmatrix} . & . \\ 1 & . \end{bmatrix} \end{array}$$

Pour rappel, elle est calculée à partir du produit $\partial = i(\partial^t r)$. Dans un premier temps, on calcule une matrice $M = \partial^t r$. Pour cela :

— on construit une matrice M temporaire, vide ;

— on parcourt les points d'entrée en colonne de r,

· on parcourt chaque maillon de la colonne de clef 0 de r. En l'occurrence, il n'y a qu'un maillon $(2, 0, -1)$. On accède au premier maillon de la colonne 2 de ∂^t, c'est-à-dire $(5, 2, 1)$, puis :

▷ on crée un maillon $(5, 0, -1 * 1)$ que l'on écrit dans M ;

▷ on passe au maillon suivant de la colonne 2 de ∂^t, c'est-à-dire $(6, 2, -1)$, puis :

▷ on crée un maillon $(6, 0, -1 * -1)$ que l'on écrit dans M ;

▷ il n'y a plus de maillon à parcourir dans la colonne 2 de ∂^t, le calcul s'arrête pour cette colonne ;

· on parcourt chaque maillon de la colonne de clef 1 de r. En l'occurrence, il n'y a qu'un maillon $(5, 1, -1)$. On accède au premier maillon de la colonne 5 de ∂^t, c'est-à-dire $(10, 5, 1)$, puis :

▷ on crée un maillon $(10, 1, -1)$ que l'on écrit dans M ;

▷ il n'y a plus de maillon à parcourir dans la colonne 5 de ∂^t, le calcul s'arrête pour cette colonne ;

· tous les points d'entrée en colonne dans r ont été parcouru, le calcul de M s'arrête.

La matrice temporaire M ainsi calculée est illustrée sur la figure 2.31. Dans un deuxième temps, on calcule $\partial = iM$. Pour cela :

— on construit une matrice ∂ temporaire ;

— on parcourt les points d'entrée en ligne de i,

· on parcourt chaque maillon de la ligne de clef 0 de i,

▷ pour le maillon $(0, 2, -1)$ de i, on parcourt la ligne 2 de M. On constate qu'il n'y a pas de maillons sur la ligne 2 de M, on ne fait rien ;

▷ pour le maillon $(0, 3, 1)$ de i, on parcourt la ligne 3 de M. On constate qu'il n'y a pas de maillons sur la ligne 3 de M, on ne fait rien ;

· on parcourt chaque maillon de la ligne de clef 1 de i,

▷ pour le maillon $(1, 5, -1)$ de i, on parcourt la ligne 5 de M. Il n'y a qu'un maillon $(5, 0, -1)$. On crée le maillon $(1, 0, -1 * -1)$ dans ∂ ;

▷ pour le maillon $(1, 9, 1)$ de i, on parcourt la ligne 9 de M. On constate qu'il n'y a pas de maillons sur la ligne 9 de M, on ne fait rien.

· tous les points d'entrée en ligne dans i on été parcouru, le calcul de M s'arrête.

La matrice temporaire ∂ ainsi calculée est illustrée sur la figure 2.32. L'équivalence homologique γ est ensuite calculée à partir de la matrice ∂ et du répertoire de cellules de B_C :

— la réduction ρ est une réduction identité. Plus précisément, cela veut dire que :

· le répertoire de cellules représentant B_{C^B} est une copie du répertoire de cellules représentant B_C, à l'exception de la `liste` $L_{B^{C^B}}$ qui le compose. Cette dernière est composée de pointeurs sur $f, g, h, \partial^b, f^S, g^S$ et h^S ;

· la matrice temporaire ∂^B est une copie de la matrice temporaire ∂ ;

· f et g sont des matrices temporaires identités ét pour lesquelles $isIdentityVariation$ vaut vrai ;

· h est une matrice temporaire nulle.

— la réduction ρ^S est calculée à partir de B_{C^B} et de ∂^B. Ce calcul dépend de l'algorithme de réduction utilisé. Notons que dans tous les cas, les opérations utilisées dans ce calcul ont une complexité limitée par la taille de B_{C^B}. À l'issue du calcul :

· un répertoire de cellule vide est créé pour représenter B_{C^S}, car B_{C^S} est une base vide ;

· f^S et g^S sont des matrices nulles ;

· h^S est une matrice temporaire composée d'un unique maillon $(0, 1, 1)$;

Les répertoires de cellules de B_{C^B} et B_{C^S} sont :

B_{C^B}	$D_{B_{C^B}}$	$L_{B_{C^B}}$
v_2	$(vrai, -, 1)$	f
e_1	$(vrai, 0, -)$	g
		h
		∂^B
		f^S
		g^S
		h^S
`first`	0	

B_{C^S}	$D_{B_{C^S}}$	$L_{B_{C^S}}$
		f^S
		g^S
		h^S
		∂^S
`first`	$-$	

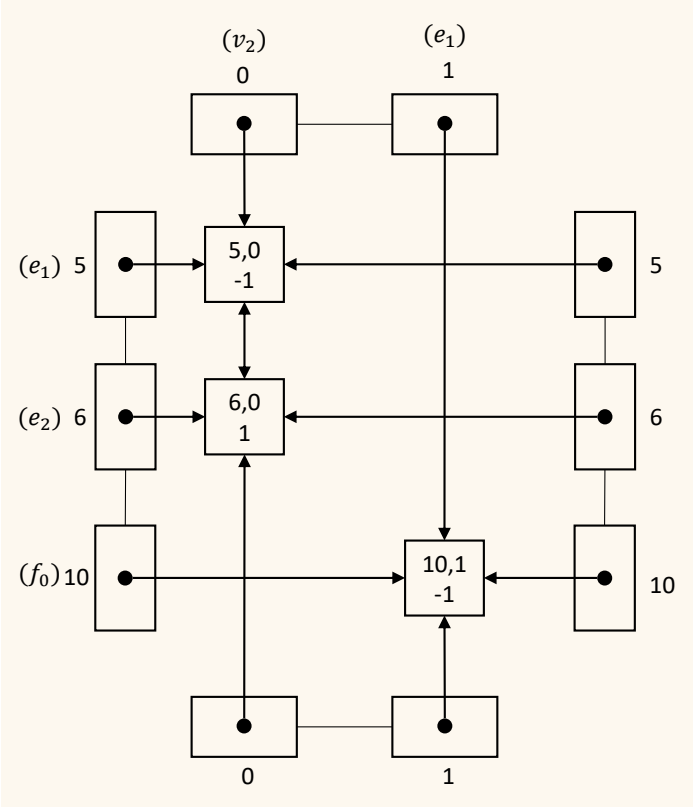


FIGURE 2.31 – La matrice temporaire $M : B_{C^t} \to B_C$, calculée à partir du produit $M = \partial^t r$. Les labels des points d'entrée correspondent aux clefs des `dictionnaires`. Entre parenthèses, les cellules que représentent les lignes/colonnes.

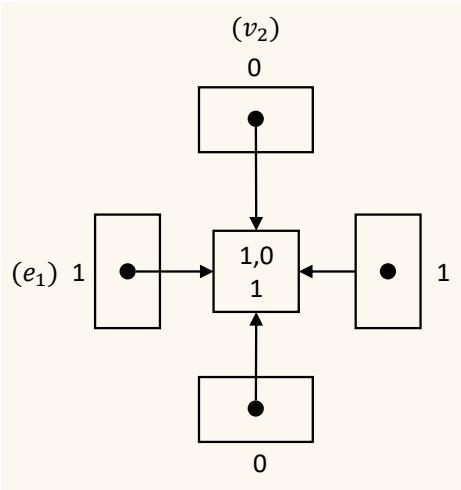


FIGURE 2.32 – La matrice temporaire $\partial : B_C \to B_C$, calculée à partir du produit $\partial = iM$. Elle est identique à la matrice $\partial^B : B_{C^B} \to B_{C^B}$. Les labels des points d'entrée correspondent aux clefs des `dictionnaires`. Entre parenthèses, les cellules que représentent les lignes/colonnes.

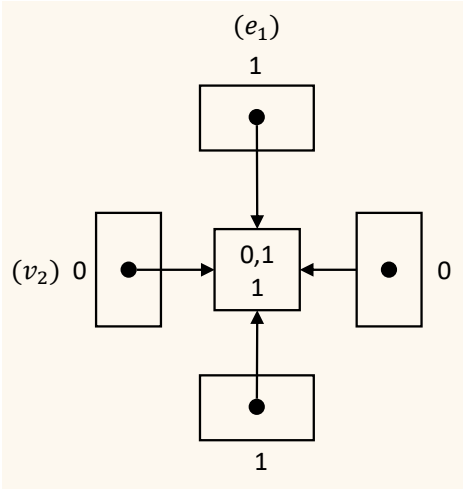


FIGURE 2.33 – La matrice temporaire $h^S : B_{C^B} \to B_{C^B}$. Les labels des points d'entrée correspondent aux clefs des `dictionnaires`. Entre parenthèses, les cellules que représentent les lignes/colonnes.

2.3.3-c Calcul de γ^{t+1}.

Une fois la SECE et γ calculées, on calcule l'équivalence homologique γ^{t+1}. Plus précisément, sont calculés, dans l'ordre :

— les matrices i^B et i^S ;

— les répertoires de cellules représentant $B_{C^{B,t+1}}$ et $B_{C^{S,t+1}}$ et les matrices $\partial^{B,t+1}$, $\partial^{S,t+1}$;

— les matrices f^{t+1}, g^{t+1}, h^{t+1}, $f^{S,t+1}$, $g^{S,t+1}$ et $h^{S,t+1}$;

— le répertoire de cellules représentant $B_{C^{t+1}}$;

Calcul des matrices i^B et i^S. Pour rappel, ces matrices sont calculées à l'aide des formules $i^B = fig^t$ et $i^S = g^S(i^B f^{S,t})$. En particulier dans l'exemple étudié :

— $i^B = i$ car $f = id$ et $g^t = id$;

— $i^S = 0_{B_{C^{S,t}}}$ car $g^S = 0_{B^{C^B}}$.

Un pointeur sur i^B et un pointeur sur i^S est ajouté respectivement dans $L_{B_{C^B}}$ et $L_{B_{C^S}}$. La figure 2.29 illustre la matrice i^B. Il n'y a rien de plus à dire sur le calcul de ces matrices.

Calcul des répertoires de cellules $B_{C^{B,t+1}}$ et $B_{C^{S,t+1}}$ et des matrices $\partial^{B,t+1}$ et $\partial^{S,t+1}$. Commençons par considérer le calcul de $\partial^{B,t+1}$ et du répertoire de cellules de $B_{C^{B,t+1}}$. Pour rappel, $\partial^{B,t+1} = \begin{pmatrix} -\partial^B & i^B \\ 0 & \partial^{B,t} \end{pmatrix}$, et $\partial^{B,t+1}$ est une modification de $\partial^{B,t}$. Rappelons également que, comme expliqué dans le paragraphe 2.3.2-e, une fusion est une opération qui implique deux étapes de calcul :

1. la mise à jour des répertoires de cellules utilisés par la matrice modifiée. En l'occurrence, le répertoire de cellules mis à jour est celui de $B_{C^{B,t}}$;
2. la "connexion" des maillons des matrices fusionnées à ceux de la matrice modifiée. En l'occurrence, les maillons de $-\partial^B$ et de i^B sont connectés à ceux de $\partial^{B,t}$.

Mise à jour du répertoire de cellules de $B_{C^{B,t}}$. Cette étape consiste à ajouter les cellules de B_{C^B} à celles de $B_{C^{B,t}}$. Pour ce faire :

— on parcourt les éléments de $D_{B_{C^B}}$. Pour chacun d'eux, on insère un nouvel élément dans $D_{B_{C^{B,t}}}$. Du fait de la montée en dimension des cellules de B_C dans $B_{C^{B,t}}$,

  · v_2 devient l'arête e_6 d'indice 12 dans $B_{C^{B,t}}$;

  · e_1 devient la face f_2 d'indice 13 dans $B_{C^{B,t}}$.

— on ajoute des colonnes/lignes vides pour représenter les points d'entrée de e_6 et f_2 dans toutes les matrices de $L_{B_{C^{B,t}}}$;

À l'issue de cette étape, le répertoire de cellules de $B_{C^{B,t}}$, qui est alors le répertoire de cellules de $B_{C^{B,t+1}}$, est :

$B_{C^{B,t+1}}$	$D_{B_{C^{B,t+1}}}$	$L_{B_{C^{B,t+1}}}$
v_0	$(vrai, -, 1)$	$\partial^{B,t}$
v_1	$(vrai, 0, 2)$	f^t
v_2	$(vrai, 1, 3)$	g^t
v_3	$(vrai, 2, 4)$	h^t
e_0	$(vrai, 3, 5)$	$h^{S,t}$
e_1	$(vrai, 4, 6)$	$f^{S,t}$
e_2	$(vrai, 5, 7)$	$g^{S,t}$
e_3	$(vrai, 6, 8)$	
e_4	$(vrai, 7, 9)$	
e_5	$(vrai, 8, 10)$	
f_0	$(vrai, 9, 11)$	
f_1	$(vrai, 10, 12)$	
e_6	$(vrai, 11, 13)$	
f_2	$(vrai, 12, -)$	
`first`	0	

Notons que, pour l'instant, en dehors de nouvelles lignes/colonnes vides, les matrices de $L_{B_{C^{B,t+1}}}$ sont inchangées (aucun nouveau maillon n'a été créé).

Connexion aux maillons de $\partial^{B,t}$. À ce stade, la matrice $\partial^{B,t}$ est composée de deux nouvelles lignes vides et de deux nouvelles colonnes vides, représentant respectivement e_6 et f_2. Il reste à connecter les maillons de $-\partial^B$ et de i^B à ceux de $\partial^{B,t}$. Pour cela :

— on parcourt les lignes de i^B, et, pour chacune d'elles, on duplique les maillons parcourus que l'on connecte à ceux de $\partial^{B,t}$. En particulier, on met à jour les indices de lignes des maillons dupliqués en remplaçant l'indice 0 par 12 et l'indice 1 par 13 ;

— on parcourt les lignes de $-\partial^B$, et, pour chacune d'elles, on duplique les maillons parcourus que l'on connecte à ceux de $\partial^{B,t}$. En particulier, on met à jour les indices de lignes et les indices de colonnes des maillons dupliqués en remplaçant l'indice 0 par 12 et l'indice 1 par 13 ;

La matrice qui résulte de la connexion des maillons de i^B et $-\partial^B$ à ceux de $\partial^{B,t}$ est illustrée sur la figure 2.34.

Répertoire de cellules de $B_{C^{S,t+1}}$ et mise à jour de $\partial^{S,t}$. Les opérations sont les mêmes que pour le répertoire de cellules de $B_{C^{B,t+1}}$ et $\partial^{B,t+1}$. En l'occurrence, dans l'exemple que nous étudions, B_{C^S} est une base vide. La matrice i^S est vide, de même que la matrice $\partial^{S,t}$. Il n'y a donc en réalité rien à calculer.

Calcul des matrices des réductions ρ^{t+1} et $\rho^{S,t+1}$. Ces matrices sont calculées à partir de fusions de matrices. Notons qu'à ce stade :

— les répertoires de cellules de $B_{C^{B,t+1}}$ et $B_{C^{S,t+1}}$ sont calculés ;

— le répertoire de cellules de B_{C^t} n'est pas encore modifié en $B_{C^{t+1}}$.

Pour simplifier, on ne considère le calcul que d'une seule matrice de ρ^t et d'une seule matrice de $\rho^{S,t}$, en l'occurrence g^{t+1} et $f^{S,t+1}$.

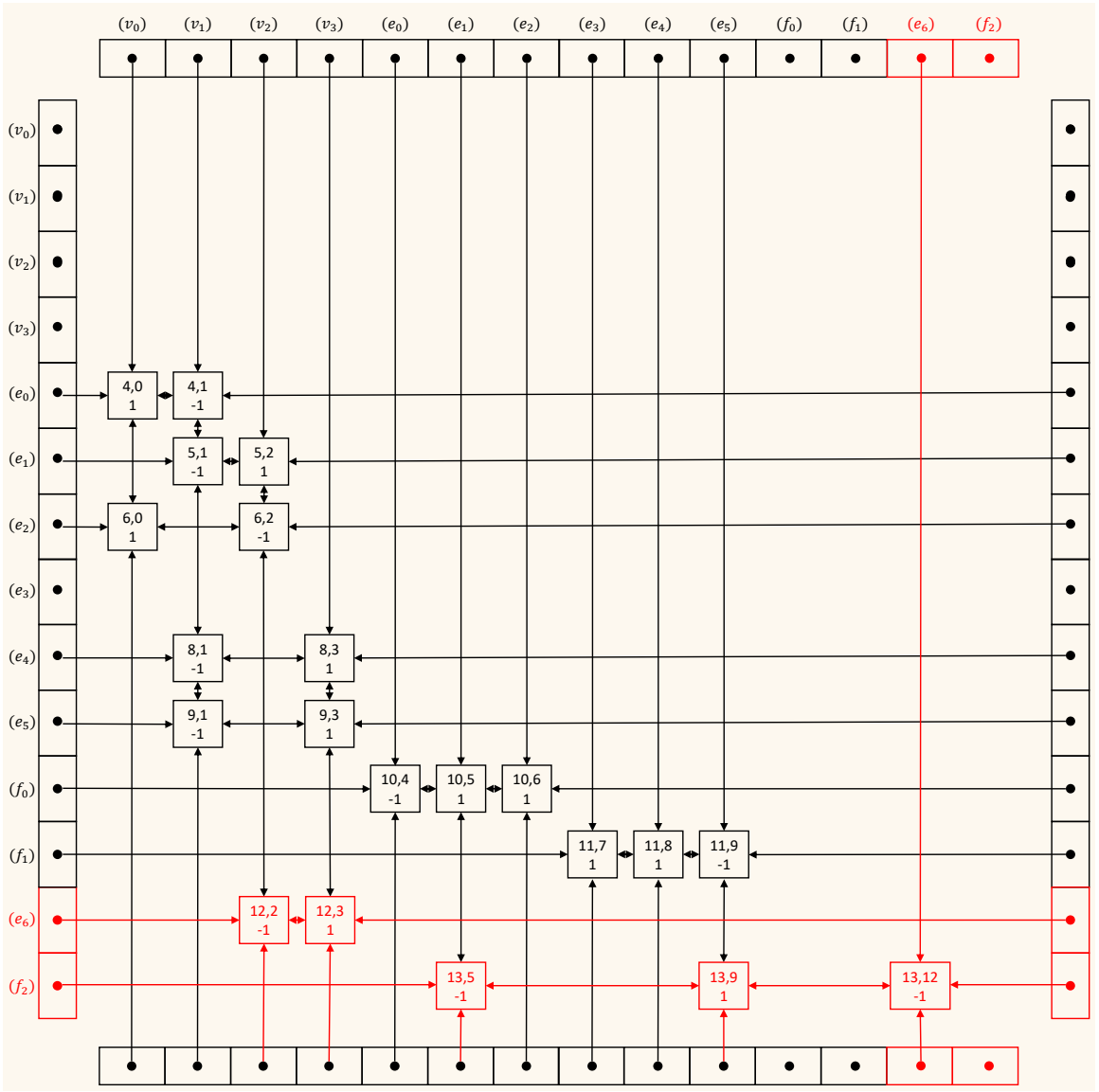


FIGURE 2.34 – La matrice maintenue $\partial^{B,t+1} : B_{C^{B,t+1}} \to B_{C^{B,t+1}}$. Les labels des points d'entrée correspondent aux clefs des `dictionnaires`. En rouge, les points d'entrée et les maillons qui ont été dupliqués et connectés. Entre parenthèses, les cellules que représentent les lignes/colonnes.

Calcul de g^{t+1}. Pour rappel, cette matrice est calculée à partir de la formule $g^{t+1} = \begin{pmatrix} -s\partial^t r g & sg^t \end{pmatrix}$. Le calcul se décompose en plusieurs temps :

- le calcul de $-\partial^t r$. Le calcul $\partial^t r$ est déjà détaillé dans le paragraphe 2.3.3-b, dans le calcul de ∂. Le produit avec -1 consiste à parcourir tous les maillons de $\partial^t r$ et à changer leur valeur ;
- la fusion de $-\partial^t r$ avec g^t. L'algorithme est identique à celui détaillé précédemment pour le calcul de $\partial^{B,t+1}$, à ceci près qu'aucun répertoire de cellules n'est modifié cette fois-ci ;

Pour terminer le calcul, il faut marquer les cellules v_2 et e_1 comme mortes dans le répertoire de cellules de B_{C^t}. Cette opération est indépendante du calcul de g^{t+1}. Elle peut être fait au

moment du calcul d'une autre matrice, voir à la fin du calcul de γ^{t+1}. Le fait de marquer ces cellules comme mortes provoque la suppression des lignes et colonnes qui les représente dans toutes les matrices de $L_{B^{C^t}}$, dont g^{t+1}. La matrice g^{t+1} est illustrée sur la figure 2.36.

Calcul de $f^{S,t+1}$. Pour rappel, cette matrice est calculée à partir de la formule $f^{S,t+1} = \begin{pmatrix} f^S & h^S i^B f^{S,t} \\ 0 & f^{S,t} \end{pmatrix}$. À ce stade, les répertoires de cellules de $B_{C^{B,t+1}}$ et de $B_{C^{S,t+1}}$ sont calculés. Le calcul se décompose en plusieurs temps :

— le calcul de $i^B f^{S,t}$ puis de $h^S(i^B f^{S,t})$. Il s'agit du même type de produit que celui de $\partial = iM$ détaillé dans le paragraphe 2.3.3-b ;

— la fusion des matrices. Les répertoires de cellules de $B_{C^{B,t+1}}$ et $B_{C^{S,t+1}}$ étant déjà calculés, cette opération se résume à la connexion des maillons de $h^S(i^B f^{S,t})$ et de f^S à ceux de $f^{S,t}$. L'algorithme est le même que celui détaillé précédemment pour la connexion de maillons dans $\partial^{B,t}$ pour le calcul $\partial^{B,t+1}$;

La matrice $f^{S,t+1}$ est illustrée sur la figure 2.35.

Calcul du répertoire de cellules de $B_{C^{t+1}}$. Ce répertoire est calculé en modifiant celui de B_{C^t}. Les cellules e_1 et v_2 sont marquées comme mortes. Plus précisément :

— l'élément d'indice 2 de $D_{B_{C^{t+1}}}$ marque la mort de v_2, c'est-à-dire qu'il devient $(faux, -, -)$, que l'entier `next` de v_1 passe à 3 et que l'entier `prev` de v_3 passe à 1 ;

— l'élément d'indice 5 de $D_{B_{C^{t+1}}}$ marque la mort de e_1, c'est-à-dire qu'il devient $(faux, -, -)$, que l'entier `next` de e_0 passe à 6 et que l'entier `prev` de e_2 passe à 4 ;

À l'issue de ces modifications, le répertoire de cellules de $B_{C^{t+1}}$ est :

$B_{C^{t+1}}$	$D_{B_{C^{t+1}}}$	$L_{B_{C^{t+1}}}$
v_0	$(vrai, -, 1)$	∂^t
v_1	$(vrai, 0, 3)$	f^t
v_2	$(faux, -, -)$	g^t
v_3	$(vrai, 1, 4)$	i
e_0	$(vrai, 3, 6)$	r
e_1	$(faux, -, -)$	
e_2	$(vrai, 4, 7)$	
e_3	$(vrai, 6, 8)$	
e_4	$(vrai, 7, 9)$	
e_5	$(vrai, 8, 10)$	
f_0	$(vrai, 9, 11)$	
f_1	$(vrai, 10, -)$	
`first`	0	

L'équivalence homologique γ^{t+1} est calculée, les matrices temporaires sont détruites.

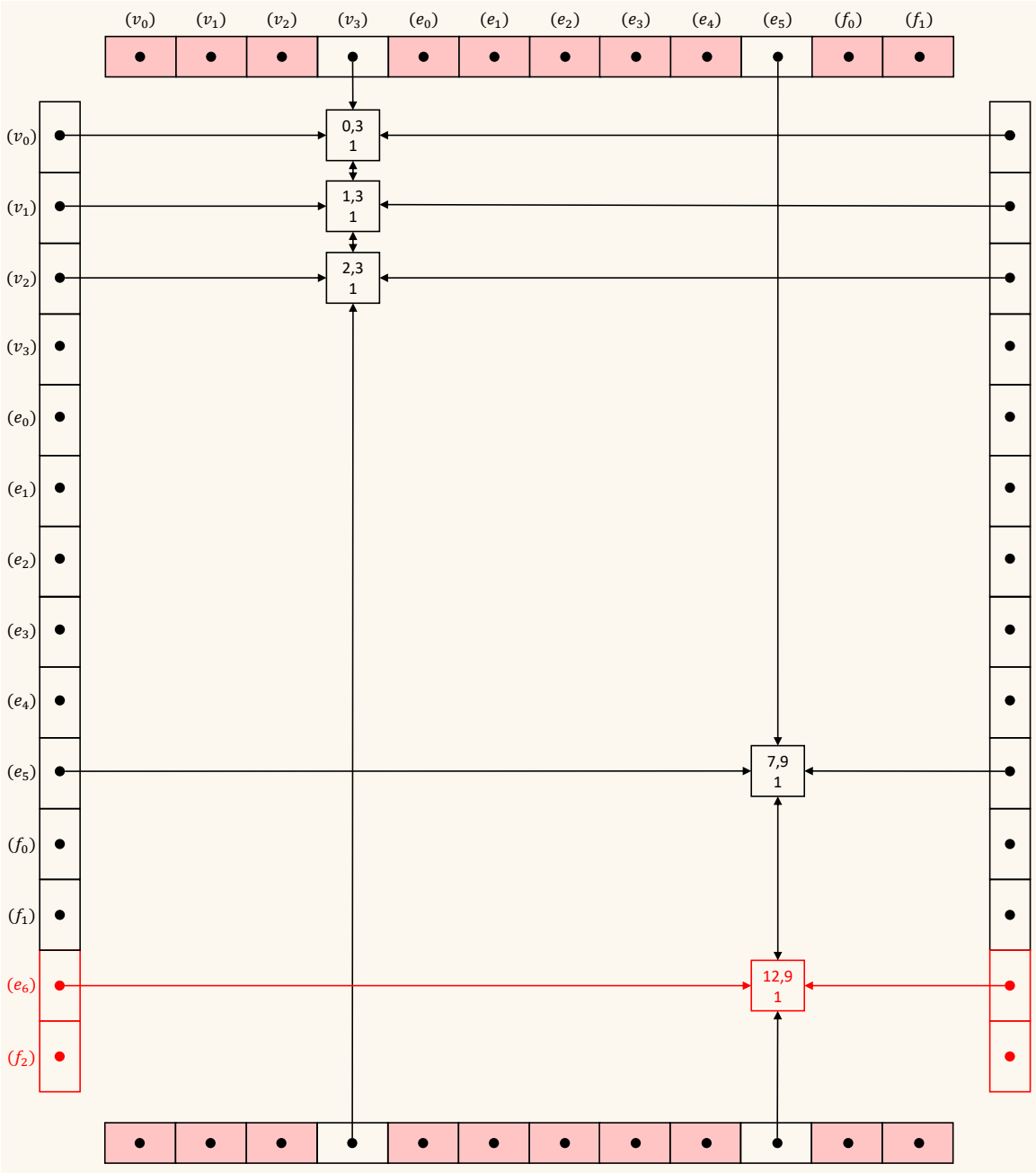


FIGURE 2.35 – La matrice maintenue $f^{S,t+1} : B_{C^{B,t+1}} \to B_{C^{S,t+1}}$. En rouge, les points d'entrée supplémentaires créés par rapport à $f^{S,t}$, et les maillons qui ont été dupliqués et connectés. Les points d'entrée avec un fond rouge indiquent que la cellule correspondante est morte dans le répertoire de cellules correspondant. Entre parenthèses, les cellules que représentent les lignes/colonnes.

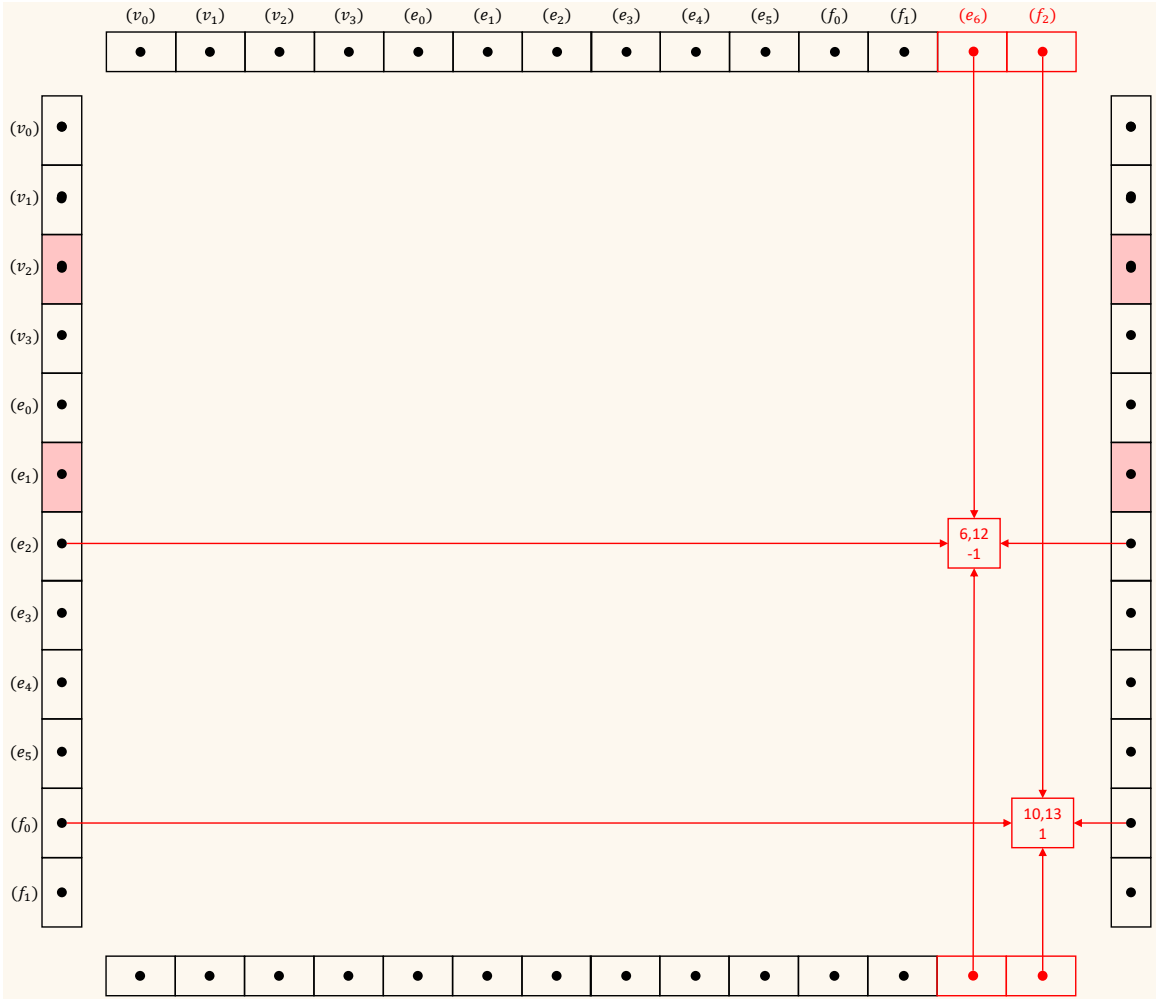


FIGURE 2.36 – La matrice maintenue g^{t+1} : $B_{C^{t+1}} \to B_{C^{B,t+1}}$. En rouge, les points d'entrée supplémentaires créés par rapport à g^t, et les maillons qui ont été dupliqués et connectés. Les points d'entrée avec un fond rouge indiquent que la cellule correspondante est morte dans le répertoire de cellules correspondant. Entre parenthèses, les cellules que représentent les lignes/colonnes.

2.3.4 Développement C++

Nous avons choisi de développer notre logiciel en utilisant la bibliothèque standard de C++ (STL). Ce paragraphe est illustré sur ce langage de programmation. L'implémentation du répertoire de cellules que nous avons réalisé se décompose en deux parties : les répertoires de cellules et les matrices. L'implémentation repose en partie sur la structure de donnée `deque` proposée par la STL.

2.3.4-a Deque.

La STL propose plusieurs conteneurs[13] adaptés à différents besoins. Le `deque` est l'un d'eux. Le nom vient de l'anglais *double-ended queue*. Intuitivement, c'est une file particulière dans laquelle l'insertion et la suppression sont définies en tête et en queue. Plus spécifiquement, en C++, cette structure garantie :

— l'insertion et la suppression en début, en temps constant ;

— l'insertion et la suppression en fin, en temps constant ;

— l'accès à un élément étant donné son indice est en temps constant.

Ces complexités sont garanties par la spécification du langage et ne dépendent pas de son implémentation (cf. [87] page 824). Pour comprendre cette structure de donnée, nous nous sommes intéressés à une de ses implémentations. Son principe est illustré sur la figure 2.37 et est le suivant :

— les données sont stockées dans plusieurs tableaux de taille B ;

— la structure conserve un vecteur (parfois appelé "tableau dynamique") T dans lequel elle conserve un pointeur vers chacun des tableaux de taille B.

L'accès à un élément se fait par le calcul de l'indice du tableau qui le contient dans T, en temps constant. En théorie, l'insertion et la suppression sont en temps constant, mais ces complexités masquent une partie de leur comportement en pratique. En particulier, on distingue 3 cas d'insertions (et des cas équivalents en suppression) :

— le premier/dernier tableau de taille B a de l'espace disponible. Dans ce cas, l'insertion est triviale ;

— le premier/dernier tableau de taille B n'a plus d'espace disponible et T a de l'espace disponible. Dans ce cas, il faut créer un nouveau tableau de taille B. Le temps est donc "constant" par rapport au nombre d'éléments du `deque`, mais un tableau de taille B doit être alloué, la constante n'est donc pas exactement la même que dans le cas de l'item précédente ;

— le dernier tableau de taille B n'a plus d'espace disponible et T n'a plus d'espace disponible. Dans ce cas, il faut créer un vecteur T avec "suffisamment" d'espace pour continuer à utiliser le `deque`. Ici, le temps est linéaire en fonction du nombre d'éléments de T.

La figure 2.38 illustre les temps mesurés pour l'insertion de $5\ 000\ 000$ d'entiers dans un `deque`. En particulier, on observe un pic de temps à certaines étapes, qui correspond aux cas où la taille de T n'est pas suffisante. Pour autant, la complexité théorique des opérations qui accompagne cette structure de donnée en font la structure la plus adaptée à nos besoins, nous avons donc choisi de l'utiliser.

13. Un conteneur est une structure de donnée qui permet de représenter et de manipuler une collection d'éléments au travers de plusieurs fonctions : `vector`, `list`, `map`, etc...

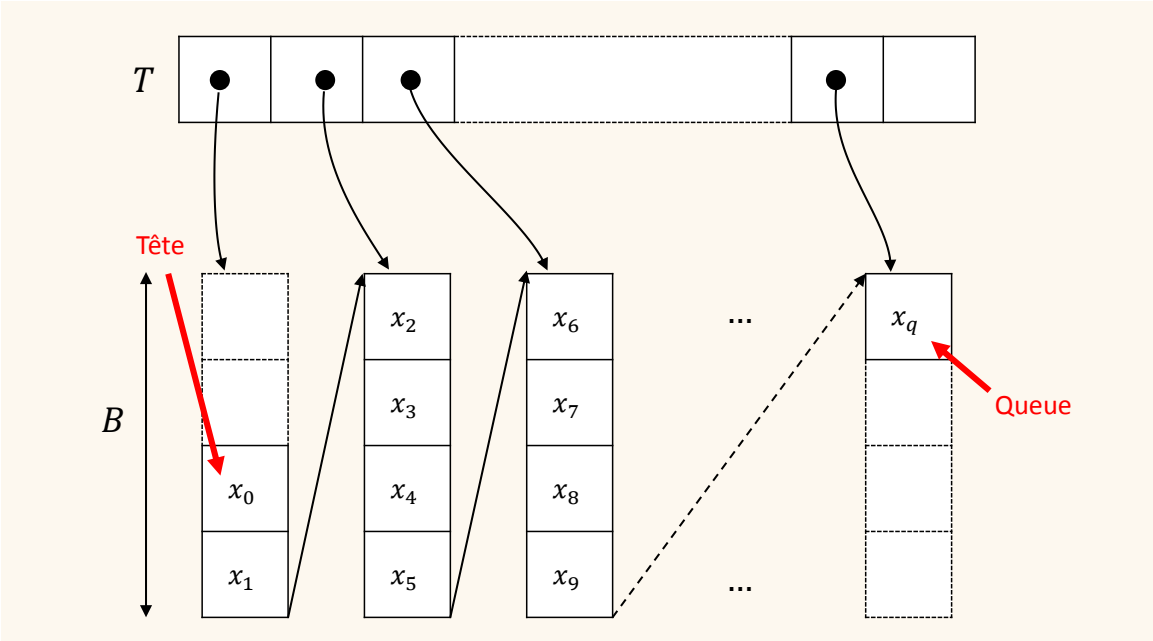


FIGURE 2.37 – Représentation d'une implémentation du deque. T est un vecteur dont certaines cases contiennent un pointeur vers un tableau de taille B. Ces derniers contiennent les données.

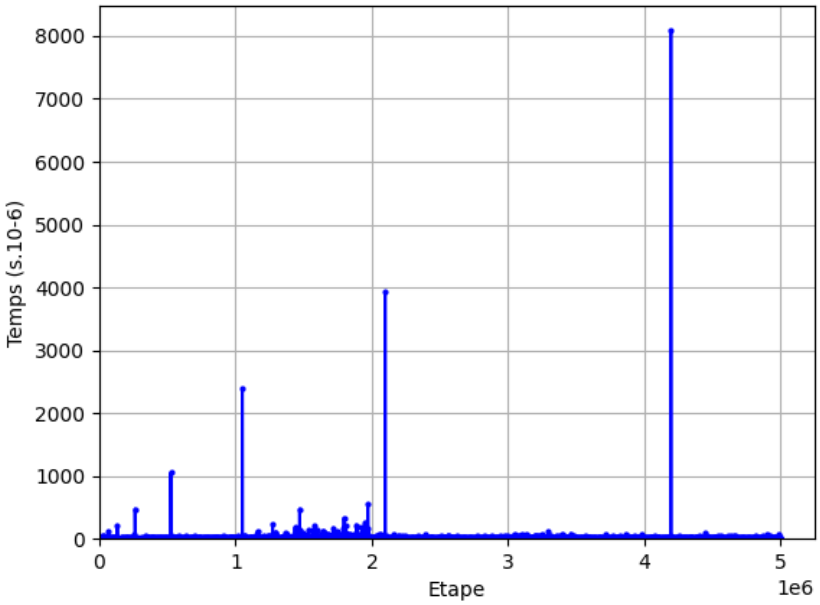


FIGURE 2.38 – Évolution du temps au fil de l'insertion de $5\ 000\ 000$ d'entiers dans un deque en C++.

2.3.4-b Répertoires de cellules.

Tout répertoire est représenté par un `deque` D, composé de triplets `(isAlive, previous, next)`. Une cellule d'une base est représentée par son indice dans D. Plus précisément, pour une cellule d'indice i :

— `isAlive` est un booléen qui représente le fait que la cellule d'indice i soit en vie ;

— `previous` est un entier qui représente l'indice de la première cellule survivante qui précède la cellule d'indice i dans la base (si elle existe) ;

— `next` est un entier qui représente l'indice de la première cellule survivante qui suit la cellule d'indice i dans la base (si elle existe).

L'intérêt des deux entiers `previous` et `next` est de permettre le parcours d'une base uniquement sur ses cellules "vivantes". Nous avons vu dans le paragraphe 2.3.1-a qu'une cellule est "tuée" dans deux cas :

— lorsqu'elle ne survit pas à une identification ;

— lorsqu'elle est réduite.

Un répertoire de cellules maintient un pointeur vers chacune des matrices dans lesquelles il est utilisé (en tant que domaine et/ou codomaine). Ces derniers sont contenus dans une `list` L, qui a le comportement d'une liste chaînée classique, à savoir :

— une insertion et une suppression en temps constant ;

— un accès à un élément étant donné son indice en temps linéaire.

2.3.4-c Représentation d'une matrice.

Les relations entre les cellules sont représentées sous la forme de maillons (i, j, v) où, pour une matrice $\phi : B_X \to B_Y$:

— i est l'indice de la cellule dans B_X ;

— j est l'indice de la cellule dans B_Y ;

— v est $\phi[i, j]$.

Les maillons sont liés entre eux par deux chaînages doubles (un "horizontal" et un "vertical"). Si une matrice est une variation de l'identité, alors les maillons de la forme (i, i, v) ne sont pas explicitement représentés, mais sont calculés : si la cellule d'indice i est vivante dans B_X et dans B_Y alors i est ajouté à l'image ou aux antécédents de i par ϕ. Chaque matrice maintient un point d'entrée permettant d'accéder au premier et au dernier maillon de chaque ligne et colonne. Plus précisément, on utilise :

— un dictionnaire (`unordered_map`) quand la matrice est temporaire. Dans ce cas, l'accès à un point d'entrée est en temps logarithmique en fonction du nombre de points d'entrée. Pour rappel, il y a autant de points d'entrée que la matrice ne compte de lignes ou colonnes **non nulles**. La construction est en temps constant ;

— un tableau dynamique (`deque`) quand la matrice est maintenue. Dans ce cas, l'accès à un point d'entrée est en temps constant. La construction est en temps linéaire en fonction du nombre d'éléments du domaine et du codomaine de la matrice (tous les points d'entrée sont représentés, y compris pour les lignes/colonnes nulles).

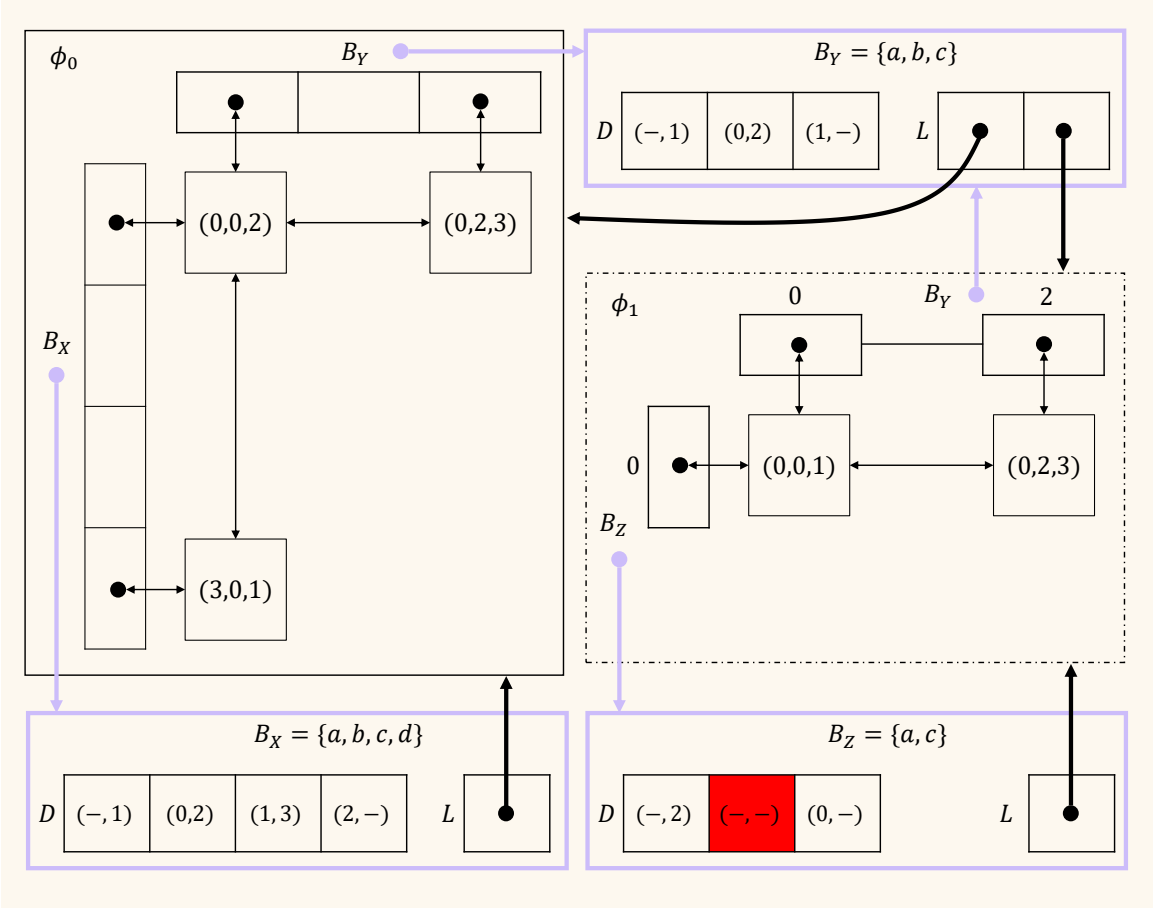


FIGURE 2.39 – Représentation de deux matrices $\phi_0 : B_X \to B_Y$ et $\phi_1 : B_Z \to B_Y$ et des répertoires de cellules de B_X, B_Y et B_Z. Les répertoires de cellules et les pointeurs des matrices vers les répertoires sont mis en évidence en violet. Les booléens `isAlive` des répertoires sont représentés par la coloration des cases de D : en rouge, les cellules mortes. Pour simplifier, seuls les pointeurs vers les premiers maillons des lignes et colonnes sont représentés. L'encadré de ϕ_1 en pointillé indique que cette matrice est temporaire.

Exemple. Soient les bases $B_X = \{a, b, c, d\}$, $B_Y = \{a, b, c\}$ et $B_Z = \{a, c\}$, et les matrices $\phi_0 : B_X \to B_Y$ et $\phi_1 : B_Z \to B_Y$ suivantes :

$$\begin{array}{c|ccc} \phi_0 & a & b & c \\ a & 2 & . & 3 \\ b & . & . & . \\ c & . & . & . \\ d & 1 & . & . \end{array} \qquad \begin{array}{c|ccc} \phi_1 & a & b & c \\ a & 2 & . & 3 \\ c & . & . & . \end{array}$$

En particulier, ϕ_0 est une matrice maintenue et ϕ_1 est une matrice temporaire, variation de l'identité. La figure 2.39 illustre l'implémentation du répertoire de cellule de B_X, B_Y et B_Z ainsi que des matrices ϕ_0 et ϕ_1. Observons que l'accès aux maillons de ϕ_0 se fait au travers d'un tableau dynamique, et l'accès aux maillons de ϕ_1 se fait au travers d'un dictionnaire.

2.4 Expérimentations

Dans un premier temps, nous présentons le processus de construction dont nous nous sommes servis pour mener une étude expérimentale de notre implémentation. Les résultats de cette étude sont présentés dans un second temps. Par ailleurs, notons qu'à ce jour, seule l'application du théorème SECE à l'identification a fait l'objet d'une étude expérimentale.

2.4.1 Processus de construction étudié : couture de deux grilles

Nous introduisons l'objet que nous utilisons pour mener l'étude expérimentale : les grilles. Ensuite, nous présentons la séquence d'identifications qui compose le processus de construction dans lequel elles évoluent, que nous nommons la "couture de grilles".

Remarque. L'annexe D propose une étude approfondie de ce processus de construction.

2.4.1-a Caractéristiques des grilles.

Les grilles sont des ensembles semi-cubiques généralisables à toute dimension et entièrement définis à partir de deux variables k et e. Intuitivement, k représente la dimension d'une grille et e permet de faire varier le nombre de cubes qui la compose.

Définition 2.4.1. *Soit $e \in \mathbb{N}$. **Une** $(1, e)$-**grille** est un ensemble semi-cubique de dimension 1, dans lequel :*

- *il y a e arêtes et $1 + e$ sommets. Les arêtes sont nommées a_y et les sommets v_x, avec $0 \leq y < e$ et $0 \leq x \leq e$;*
- *$a_y d_{1,0} = v_y$ et $a_y d_{1,1} = v_{y+1}$;*

Définition 2.4.2. *Une (k, e)-**grille** est le produit cartésien de k $(1, e)$-grilles, avec $k, e \in \mathbb{N}$.*

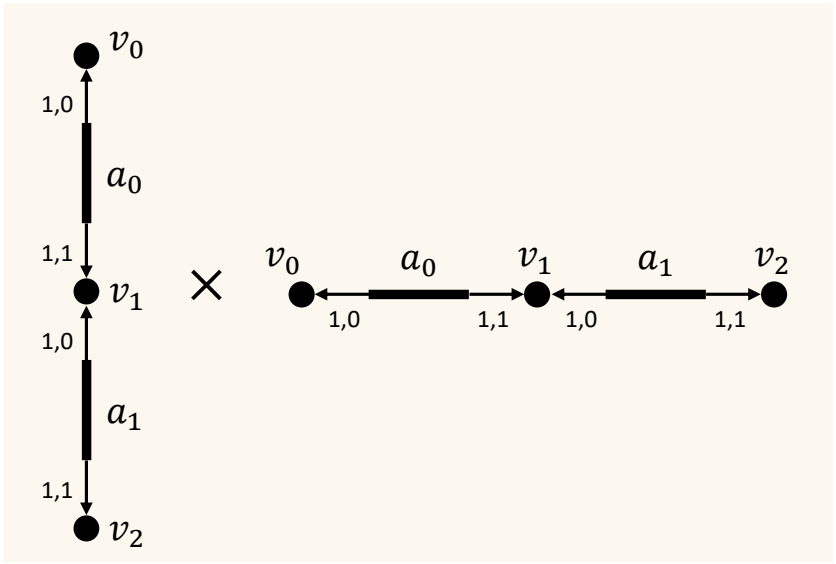
Nous avons choisi d'utiliser les grilles car elles permettent d'étudier facilement une grande diversité de cas au travers de seulement 2 variables. Par exemple, on peut fixer k et faire évoluer e pour étudier l'évolution d'une ressource de calcul (le temps, l'espace...) en fonction du nombre de cubes qui compose une grille. Par ailleurs, notons qu'en pratique nous manipulons le complexe de chaînes induit par une grille et non l'ensemble semi-cubique qui la représente.

Exemple. La figure 2.40 illustre le produit cartésien à l'origine d'une $(2, 2)$-grille et le complexe de chaînes induit par l'ensemble semi-cubique qui en résulte. En particulier, observons sur la figure 2.40a que chaque $(1, 2)$-grille du produit est telle que :

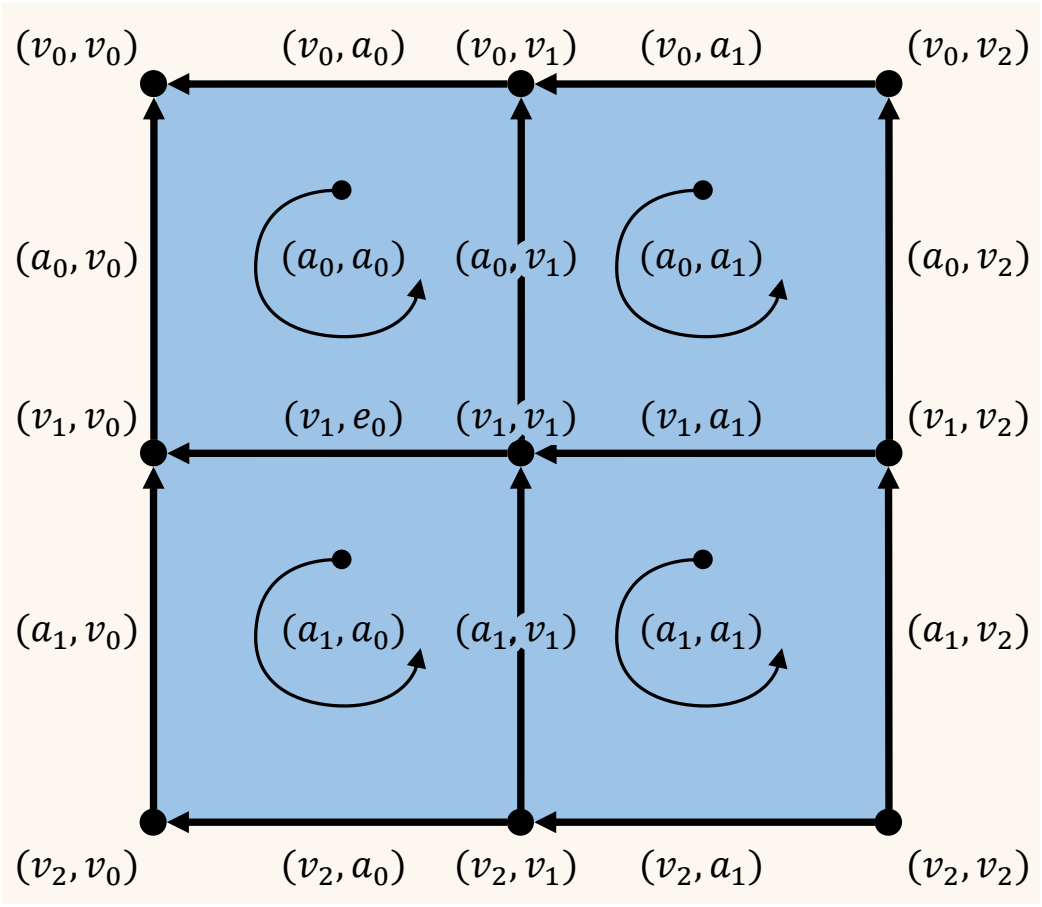
- $a_0 d_{1,0} = v_0$ et $a_0 d_{1,1} = v_1$;
- $a_1 d_{1,1} = v_1$ et $a_1 d_{1,1} = v_2$.

Le bord des p-cubes découle de la définition du produit cartésien (cf. définition 1.2.12). En particulier, on déduit de la figure 2.40b que, pour le 2-cubes (a_0, a_0) :

- $(a_0, a_0)d_{1,0} = (a_0 d_{1,0}, a_0) = (v_0, a_0)$ et $(a_0, a_0)d_{1,1} = (a_0 d_{1,1}, a_0) = (v_1, a_0)$;
- $(a_0, a_0)d_{2,0} = (a_0, a_0 d_{1,0}) = (a_0, v_0)$ et $(a_0, a_0)d_{2,1} = (a_0, a_0 d_{1,1}) = (a_0, v_1)$.



(a)



(b)

FIGURE 2.40 – Le produit cartésien de deux $(1, 2)$-grilles, à l'origine d'une $(2, 2)$-grille, et le complexe de chaînes induit par la $(2, 2)$-grille. (a) Le produit cartésien. (b) Le complexe de chaînes.

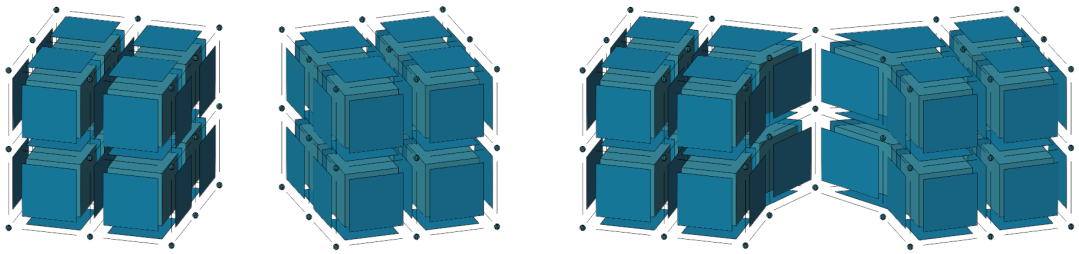
2.4.1-b Couture de deux grilles.

La couture de deux grilles est un processus de construction dans lequel on colle deux (k, e)-grilles le long d'une (n, e)-grille, avec $n < k$. Pour cela, on applique une séquence d'identifications entièrement définies à partir des 3 variables k, n et e. Au total, le processus est composé de $(e^n + 1)$ étapes de construction où :

— e^n étapes consistent à identifier un même *motif* ;

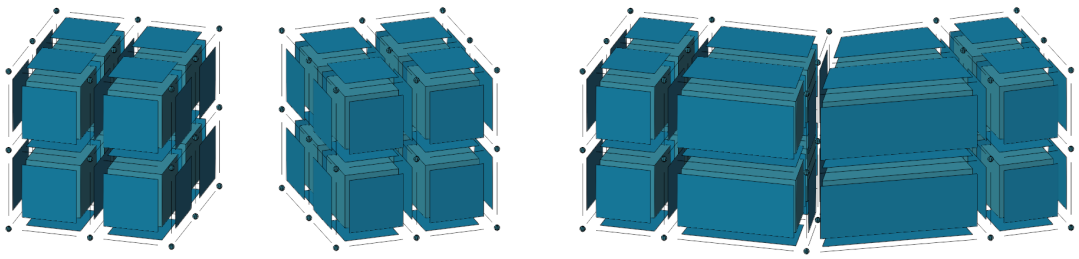
— la première étape consiste à identifier le bord de l'ensemble des motifs.

Les figures 2.41 et 2.42 illustrent la première et la dernière étape de la couture de deux $(3, 2)$-grilles le long, respectivement, d'une $(1, e)$-grille et d'une $(2, e)$-grille.



(a) Étape $t = 0$. (b) Étape $t = 3$.

FIGURE 2.41 – Couture de deux $(3, 2)$-grilles le long d'une $(1, 2)$-grille. (a) La première étape de construction. (b) La dernière étape de construction.



(a) Étape $t = 0$. (b) Étape $t = 5$.

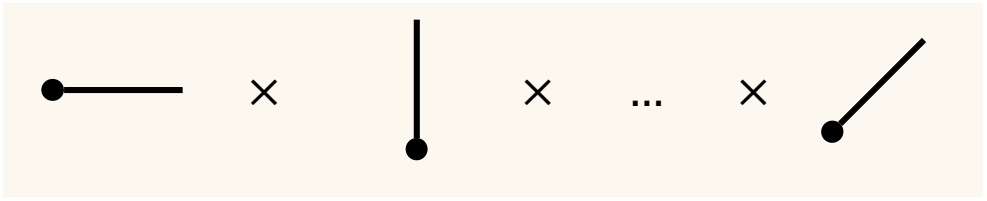
FIGURE 2.42 – Couture de deux $(3, 2)$-grilles le long d'une $(2, 2)$-grille. (a) La première étape de construction. (b) La dernière étape de construction.

Pour simplifier cette section, nous excluons la première étape ($t = 0$) de l'étude et nous nous focalisons sur les identifications de *motifs*. Un *motif* est une partie du produit cartésien à l'origine d'une (k, e)-grille, composée de 2^n cubes. Il est construit en prenant une arête et un sommet de son bord dans n $(1, e)$-grille à l'origine d'une (k, e)-grille. Dans une couture de grilles, l'identification de passage de t à $t + 1$ pour $t \neq 0$ consiste simplement à identifier un même motif entre deux (k, e)-grilles distinctes.

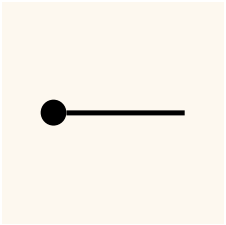
Définition 2.4.3. *Un n-**motif** est une partie du produit cartésien à l'origine d'une (k, e)-grille, composée de :*

- *de n arêtes a_y et des n sommets v_{y+1} de leur bord choisis parmi les n premières $(1, e)$-grilles pour y tel que $0 \leq y < e$;*
- *de $n - k$ sommets v_0 choisis parmi les $n - k$ dernières $(1, e)$-grilles.*

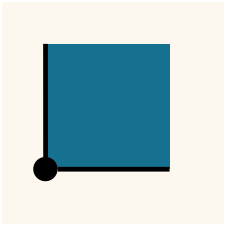
La figure 2.43 illustre des motifs pour plusieurs valeurs de n. Nous avons choisi d'utiliser ces motifs car ils se généralisent à toute dimension et permettent de couvrir une grande diversité de cas au travers des 2 variables n et e.



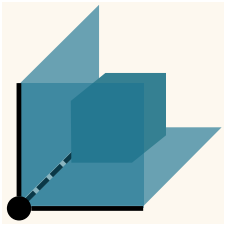
(a)



(b) $n = 1$



(c) $n = 2$



(d) $n = 3$

FIGURE 2.43 – Des motifs de différentes dimensions. (a) La partie du produit cartésien à l'origine d'une (k, e)-grille utilisée pour calculer un motif. (b,c,d) Des motifs pour différentes valeurs de n.

2.4.2 Résultats expérimentaux

Nous étudions expérimentalement l'implémentation de l'application du théorème SECE à l'identification, en utilisant l'implémentation décrite en sous-section 2.3. Le processus de construction étudié est la "couture de deux grilles". Nous distinguons le cas où les générateurs d'homologie sont calculés et le cas où ils ne le sont pas.

Remarque. L'étude expérimentale se limite pour l'instant à l'opération d'identification. L'implémentation utilisée pour générer les résultats de l'ensemble de cette sous-section est disponible à l'adresse `https://gitlab.xlim.fr/hominc/HomInc`. De plus, les mesures de cette sous-section ont été prises sur une machine équipée d'un Intel Xeon Gold 5118 @2.30GHz et de $137Go$ de RAM. Rappelons que l'utilisation de la mémoire par un processus dépend essentiellement du système dans lequel le processus évolue. Nous concernant, le système utilisé est Windows 10. En particulier, nous mesurons la "mémoire" à l'aide de la fonction `GetProcessMemoryInfo` de l'entête C++ `Windows.h`. Les valeurs présentées correspondent au champ `PeakWorkingSetSize` qui fait partie du résultat de cette fonction. Ce champ représente la plus grande quantité de mémoire physique utilisée par un processus depuis qu'il existe.

2.4.2-a Identification sans calcul des générateurs d'homologie

Pour rappel, lorsque les générateurs d'homologie ne sont pas calculés, une partie des matrices de l'équivalence homologique maintenue ne sont pas calculées (cf. sous-section 2.1.5). Les résultats théoriques sont que :

— la complexité en temps du calcul de γ^{t+1} dépend du nombre de cellules impactées par l'opération de passage de t à $t+1$, et de leur étoile "directe" [14] ;

— la complexité en espace de γ^{t+1} est égale à la complexité en espace de $\gamma^{t=0}$ à laquelle s'ajoute la somme des variations en espace des étapes précédentes. La variation d'une étape à l'autre dépend du motif identifié, et plus particulièrement de son nombre de cubes.

Nous montrons que ces résultats théoriques se confirment expérimentalement sur la couture de grilles. Les cas étudiés sont, pour $e \in \mathbb{N}$:

— la couture de deux $(3, e)$-grilles le long d'une $(2, e)$-grille ;

— la couture de deux $(4, e)$-grilles le long d'une $(3, e)$-grille ;

— la couture de deux $(5, e)$-grilles le long d'une $(4, e)$-grille.

Temps. Les mesures sont en microsecondes ($s.10^{-6}$). Les résultats sont exprimés sur une échelle de temps variable adaptée aux mesures observées. Les temps mesurés sont illustrés sur les figures 2.44, 2.45 et 2.46.

Évolution du temps au fil des étapes. Intéressons-nous aux courbes d'évolution du temps au fil des étapes. Nous observons une oscillation du temps. Ce phénomène est dû aux différents cas d'insertion et de suppression du `deque` (cf sous-section 2.3.4). Pour autant, notons que la tendance de la courbe est constante. Dit autrement, le **temps de calcul est constant au fil des étapes de construction**. En particulier, observons que :

14. L'étoile directe d'une p-cellule est son étoile restreinte aux $(p+1)$-cellules.

— pour k et n fixées, le fait d'augmenter la valeur de e n'influe pas sur le temps de calcul moyen. Dit autrement, **le nombre de cellules des deux (k,e)-grilles, sur lesquelles sont appliquées les identifications, n'influe pas sur le temps de calcul moyen**. Ainsi,

  · sur les figures 2.44a et 2.44b, on observe un temps moyen d'environ $500\mu s$ pour $e = 25$ et $e = 110$;

  · sur les figures 2.45a et 2.45b, on observe un temps moyen d'environ $1200\mu s$ pour $e = 10$ et $e = 35$;

  · sur les figures 2.46a et 2.46b, on observe un temps moyen d'environ $2100\mu s$ pour $e = 8$ et $e = 15$.

— plus la valeur de n est grande, plus le temps de calcul moyen est élevé. En effet, nous constatons que :

  · pour $n = 2$, le temps moyen est d'environ $500\mu s$;

  · pour $n = 3$, le temps moyen est d'environ $1200\mu s$;

  · pour $n = 5$, le temps moyen est d'environ $2100\mu s$.

  Cette observation fait sens car augmenter n revient à augmenter le nombre de cellules identifiées à chaque étape de construction. Dit autrement, étant donnée l'équivalence homologique d'une étape t, **le nombre de cellules identifiées lors du passage de t à $t+1$ influe sur le temps de calcul de l'équivalence homologique à $t+1$.**

Partage du temps de calcul entre les matrices. Intéressons nous aux graphiques de partage du temps entre les matrices. Les données représentées sont :

— `c`, qui représente le calcul des générateurs du complexe de chaînes (C, ∂) à partir de l'identification ;

— `gamma`, qui représente le calcul de l'équivalence homologique γ ;

— `extra`, qui représente le fait de tuer les générateurs non-survivants dans les répertoires de cellules ;

— `fSt`, `gt`, qui représentent respectivement le calcul de $f^{S,t}$ et g^t ;

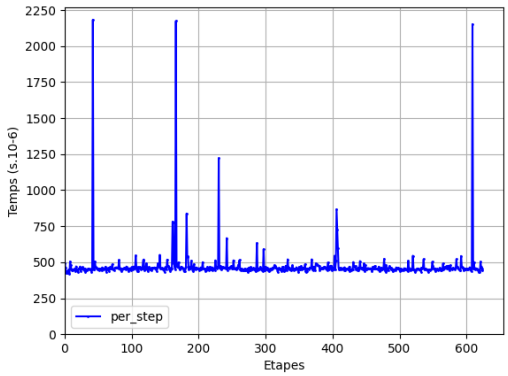
— `i`, `iB`, `iS` et `r`, qui représentent respectivement le calcul de i, i^B, i^S et r ;

— `partialt`, `partialBt` et `partialSt`, qui représentent respectivement le calcul de $(C^{t+1}, \partial^{t+1})$, $(C^{B,t+1}, \partial^{B,t+1})$ et $(C^{S,t+1}, \partial^{S,t+1})$.

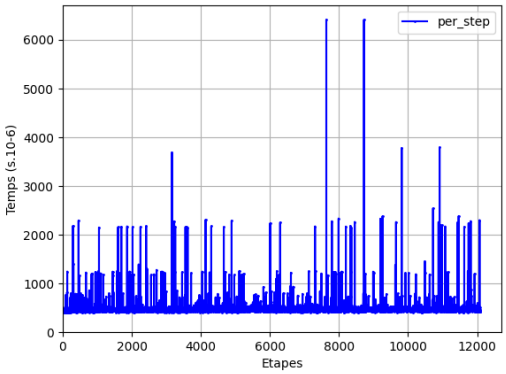
On observe sur ces graphiques que, peu importe le cas étudié :

— les données les plus coûteuses sont `gamma` et `gt`. Il n'y a pas d'interprétation particulière de cette observation si ce n'est que le fait que `gamma` soit la donnée la plus coûteuse montre que le nombre de cellules de la grille n'influe pas sur le temps de calcul car, pour rappel, γ est entièrement définie à partir des cellules non-survivantes (cf. paragraphe 2.1.2-b) ;

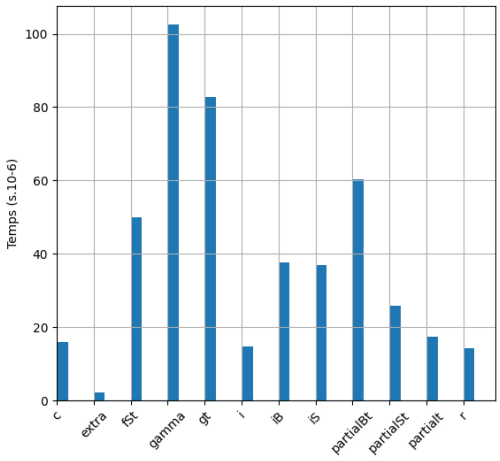
— plus n est grand, plus le temps de calcul de `gt` se rapproche de celui de γ et représente une part importante du temps total. Ceci fait sens car une partie de la modification de g^t demande de calculer le produit $s\partial^t rg$. Or, le temps de calcul de ce produit dépend du nombre de cellules dans l'étoile "directe" des cellules non-survivantes à l'identification de passage de t à $t+1$, et ce nombre augmente avec n.



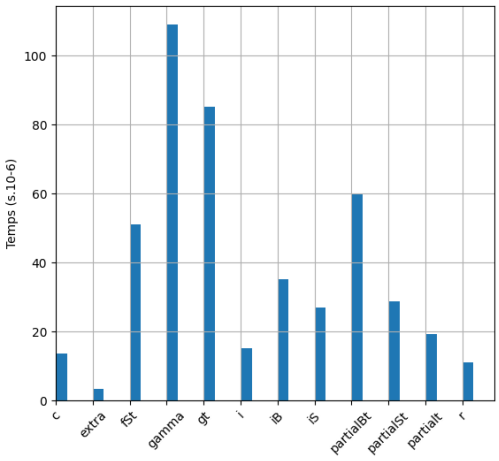
(a) Évolution du temps au fil des étapes ($k3e25n2$)



(b) Évolution du temps au fil des étapes ($k3e110n2$)

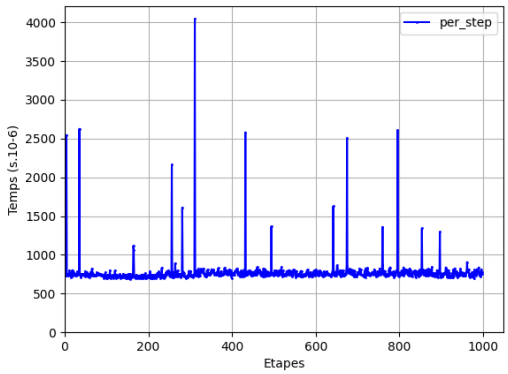


(c) Partage du temps entre les matrices ($k3e25n2$)

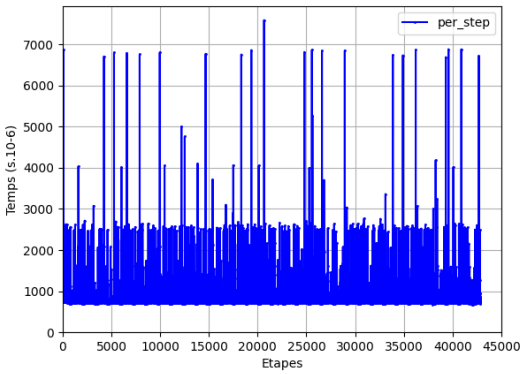


(d) Partage du temps entre les matrices ($k3e110n2$)

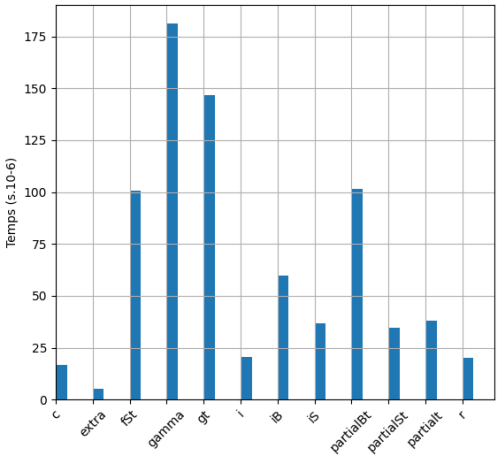
FIGURE 2.44 – Résultats en temps pour la couture de deux $(3, e)$-grilles le long d'une $(2, e)$-grille. (a,c) $e = 25$, soit 265 302 cellules à $t = 0$. (b, d) $e = 110$, soit 21 587 722 cellules à $t = 0$.



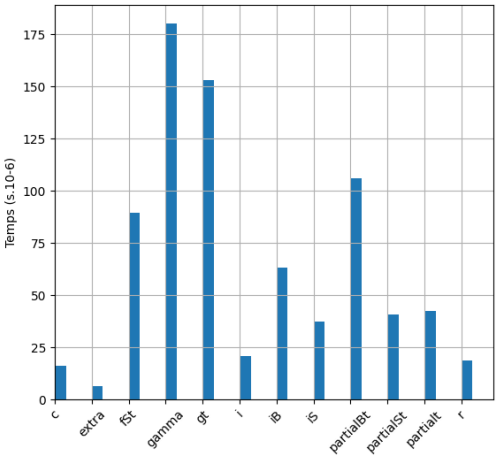
(a) Evolution du temps au fil des étapes ($k4e10n3$)



(b) Evolution du temps au fil des étapes ($k4e35n3$)

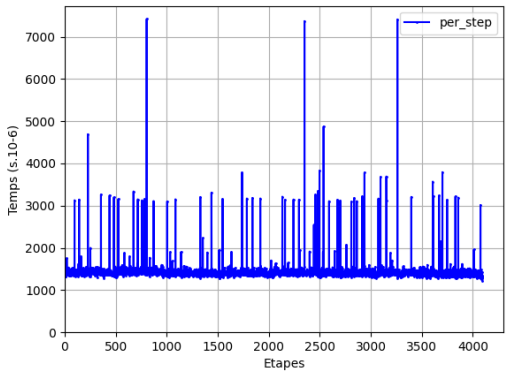


(c) Partage du temps entre les matrices ($k4e10n3$)

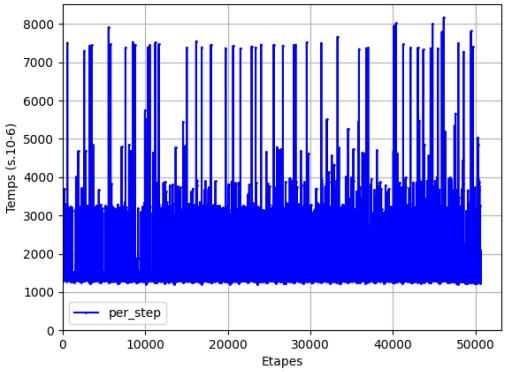


(d) Partage du temps entre les matrices ($k4e35n3$)

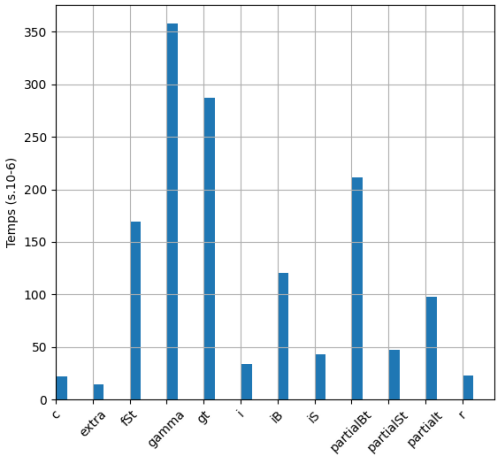
FIGURE 2.45 – Résultats en temps pour la couture de deux $(4, e)$-grilles le long d'une $(3, e)$-grille. (a, c) $e = 10$ soit $388\,962$ cellules à $t = 0$. (b, d) $e = 35$ soit $50\,823\,362$ cellules à $t = 0$.



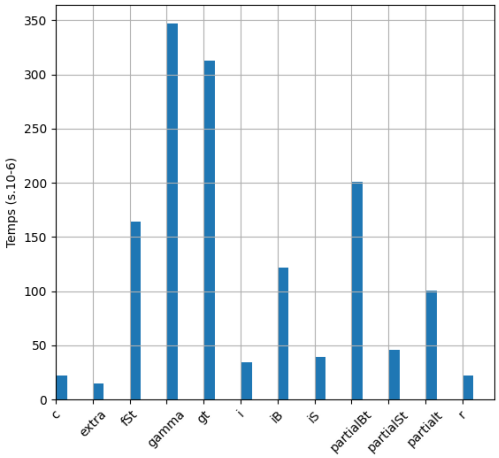
(a) Evolution du temps au fil des étapes ($k5e8n4$)



(b) Evolution du temps au fil des étapes ($k5e15n4$)



(c) Partage du temps entre les matrices ($k5e8n4$)

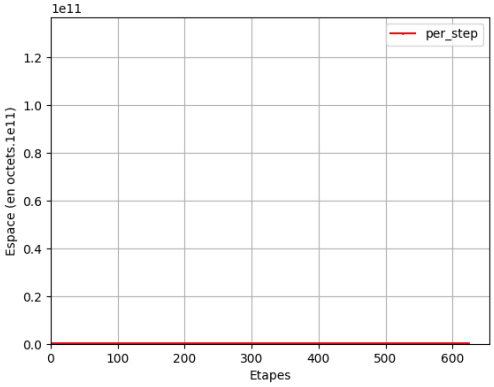


(d) Partage du temps entre les matrices ($k5e15n4$)

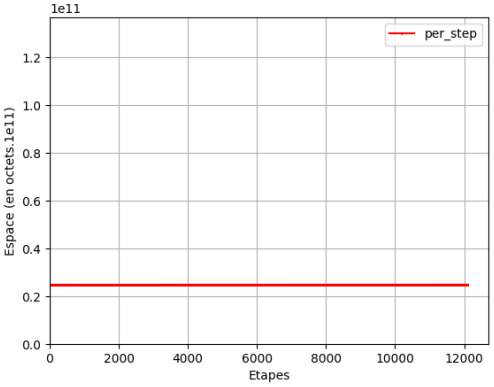
FIGURE 2.46 – Résultats en temps pour la couture de deux $(5, e)$-grilles le long d'une $(4, e)$-grille. (a,c) $e = 8$ soit $2\,839\,714$ cellules à $t = 0$. (b, d) $e = 15$ soit $57\,258\,302$ cellules à $t = 0$.

Espace. Les mesures sont en octets. Les résultats sont exprimés sur une échelle de 0 à $137Go$. L'espace mesuré est illustré sur les figures 2.47, 2.48 et 2.49. Sur les courbes, nous observons que l'utilisation de l'espace est constante au fil des étapes du processus de construction. En réalité, il y a une très légère augmentation de la consommation mémoire du processus entre la première et la dernière étape de construction (moins de $1Go$ supplémentaire). Cette augmentation fait sens car, à chaque étape, la quantité de données ajoutées dans γ^t est "petite", comparée à la taille de γ^t en mémoire. Ces résultats traduisent le fait que le coût en espace d'une étape t donnée est égal au coût de l'étape $t = 0$ auquel s'ajoute la somme des variations en espace des étapes précédentes.

Remarque. Les mesures ne sont représentées que sous la forme d'évolution au fil des étapes de construction. En particulier, nous ne présentons pas de graphiques de répartition de l'espace entre les matrices car nos mesures représentent une quantité de mémoire physique. Or, les allocations et désallocation de mémoire physique dépendent en grande partie de l'état du système au moment de la mesure. Plus précisément, la stratégie utilisée par Windows 10 est "paresseuse", c'est-à-dire que la mémoire physique attribuée à un processus lui reste attribuée tant qu'il existe et qu'aucun autre processus n'a besoin de mémoire physique. Ce fonctionnement permet notamment d'éviter des désallocations/allocations superflues, mais rend difficilement interprétable une prise de mesure fine de l'espace mémoire physique utilisé par un processus. Pour cette raison, nous avons choisi de mesurer le pic mémoire du processus depuis sa création, à chaque étape.

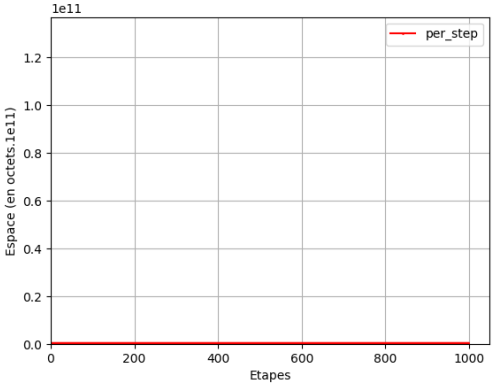


(a) Evolution de l'espace au fil des étapes $(k3e25n2)$

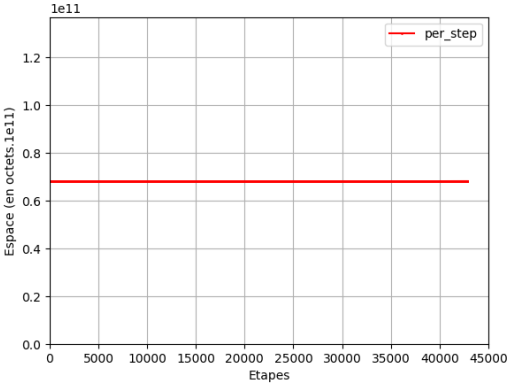


(b) Evolution de l'espace au fil des étapes $(k3e110n2)$

FIGURE 2.47 – Résultats en espace pour la couture de deux $(3, e)$-grilles le long d'une $(2, e)$-grille. (a) $e = 25$, soit $265\ 302$ cellules à $t = 0$. (b) $e = 110$, soit $21\ 587\ 722$ cellules à $t = 0$.

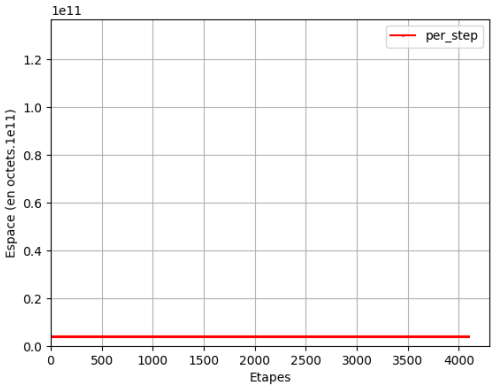


(a) Evolution de l'espace au fil des étapes $(k4e10n3)$

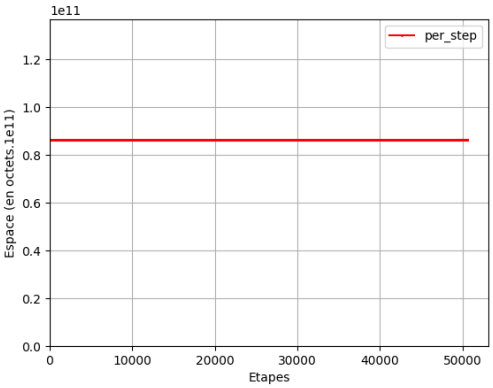


(b) Evolution de l'espace au fil des étapes $(k4e35n3)$

FIGURE 2.48 – Résultats en espace pour la couture de deux $(4, e)$-grilles le long d'une $(3, e)$-grille. (a) $e = 10$ soit $388\ 962$ cellules à $t = 0$. (b) $e = 35$ soit $50\ 823\ 362$ cellules à $t = 0$.



(a) Evolution de l'espace au fil des étapes $(k5e8n4)$



(b) Evolution de l'espace au fil des étapes $(k5e15n4)$

FIGURE 2.49 – Résultats en espace pour la couture de deux $(5, e)$-grilles le long d'une $(4, e)$-grille. (a) $e = 8$ soit $2\ 839\ 714$ cellules à $t = 0$. (b) $e = 15$ soit $57\ 258\ 302$ cellules à $t = 0$.

2.4.2-b Identification avec calcul des générateurs d'homologie

Pour rappel, lorsque les générateurs d'homologie sont calculés, toutes les matrices de l'équivalence homologique maintenue sont calculées (cf. sous-section 2.1.5). Les résultats théoriques sont les mêmes que lorsque les générateurs d'homologie ne sont pas calculés, sauf pour le calcul de $h^{S,t+1}$, qui, dans le pire des cas, est lié au nombre de cellules de la structure topologique. Les résultats théoriques se confirment expérimentalement sur la couture de grilles. Nous montrons que ces résultats théoriques se confirment expérimentalement sur la couture de grilles. Les cas étudiés sont les mêmes que ceux étudiés sans calcul des générateurs d'homologie.

Remarque. La même machine et les mêmes fonctions de mesures sont utilisées dans ce paragraphe que celles utilisées dans le paragraphe 2.4.2-a.

Temps. Les mesures sont en microsecondes ($s.10^{-6}$). Les résultats sont exprimés sur une échelle de temps variable adaptée aux mesures observées. Les temps mesurés sont illustrés sur les figures 2.51, 2.52 et 2.53.

Évolution du temps au fil des étapes. On observe que le **temps moyen augmente au fil des étapes de construction**. Nous expliquons cette évolution par le calcul de $h^{S,t+1}$. En effet, cette matrice est calculée à l'aide de la formule $h^{S,t+1} = \begin{pmatrix} -h^S & h^S i^B h^{S,t} \\ 0 & h^{S,t} \end{pmatrix}$. Or, le temps de calcul du produit $h^S i^B h^{S,t}$ et de sa fusion varient en fonction des étapes. Pour comprendre cette variation, il faut étudier plus en détail la réduction que nous appliquons aux grilles [15]. Intuitivement, elle traduit le fait que le nombre de cellules dans l'image d'une cellule de C^B par $h^S i^B h^{S,t}$ dépend de "l'endroit" où le motif est identifié dans les deux (k, e)-grilles. La figure 2.50 illustre ce phénomène pour les sommets d'une $(3, 2)$-grille : il y a plus ou moins d'arêtes dans l'image d'un sommet par $h^{S,t}$ selon "l'endroit" où le sommet est choisi. Cette variation est particulièrement visible sur les graphiques d'évolution du temps des figures 2.51a et 2.51b où l'on observe que le temps par étape oscille.

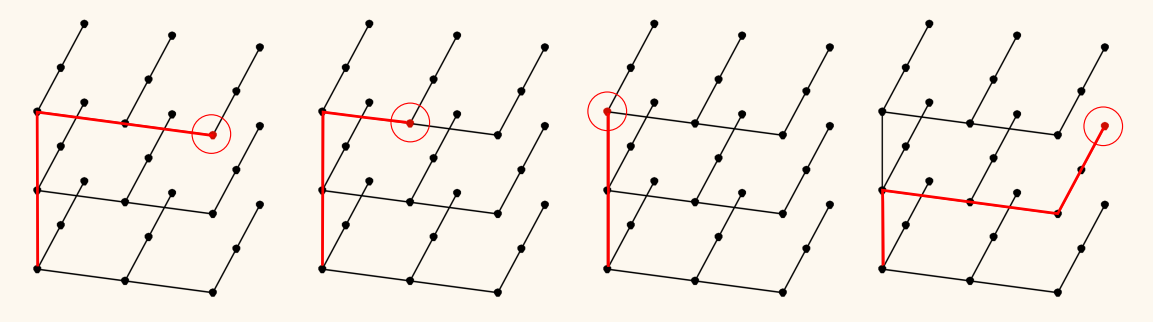


FIGURE 2.50 – L'image d'un somme d'une $(3, 2)$-grille par $h^{S,t}$ pour différents sommets. En rouge, les arêtes dans l'image du sommet encerclé.

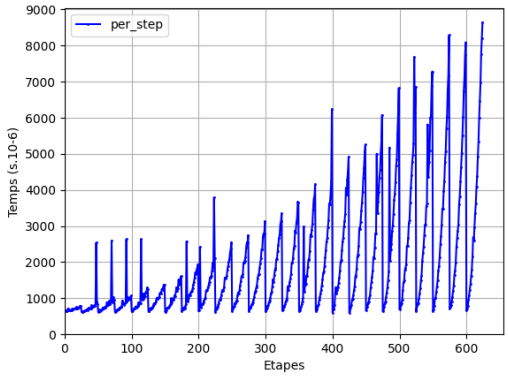
Partage du temps de calcul entre les matrices. Les graphiques de partage du temps reprennent les mêmes données que celles du paragraphe 2.4.2-a, auxquelles s'ajoute :

— `gSt` et `ft` qui représentent respectivement le temps de calcul de $g^{S,t+1}$ et f^{t+1} ;

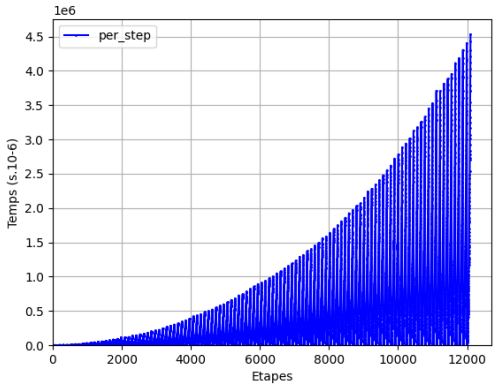
15. L'étude de la réduction d'une grille est disponible en annexe D.2.

— `ht` et `hSt` qui représentent respectivement le temps de calcul de h^{t+1} et $h^{S,t+1}$.

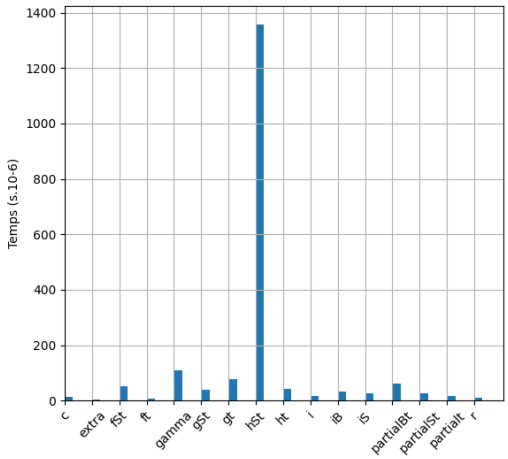
Sur ces graphiques, on observe que le calcul de `hSt` est largement prédominant. Cela confirme que l'oscillation du temps par étape est dû au fait que le nombre de cellules de $(C^{B,t}, \partial^{B,t})$ dans l'image d'une cellule de $(C^{B,t}, \partial^{B,t})$ est variable en fonction des cellules considérées.



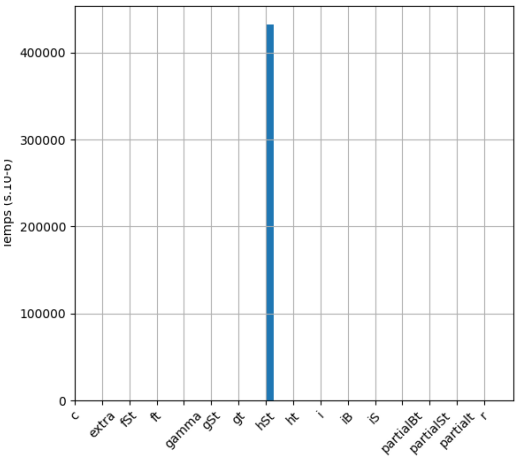
(a) Évolution du temps au fil des étapes $(k3e25n2g)$



(b) Évolution du temps au fil des étapes $(k3e110n2g)$

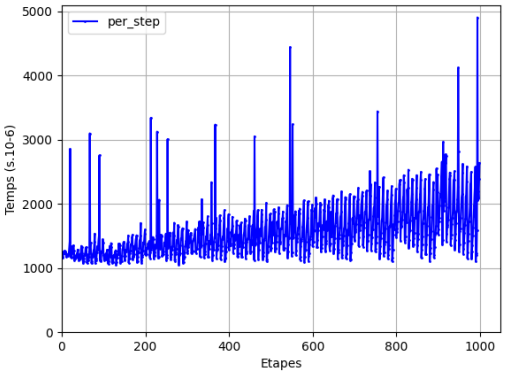


(c) Partage du temps entre les matrices $(k3e25n2g)$

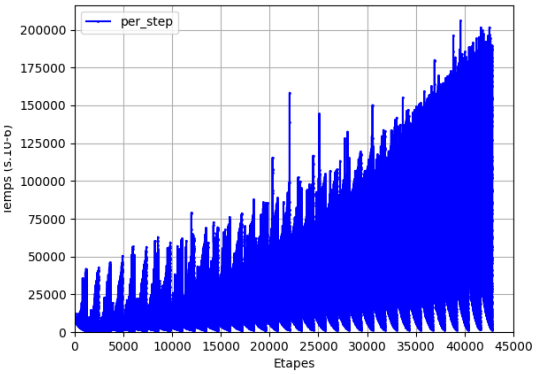


(d) Partage du temps entre les matrices $(k3e110n2g)$

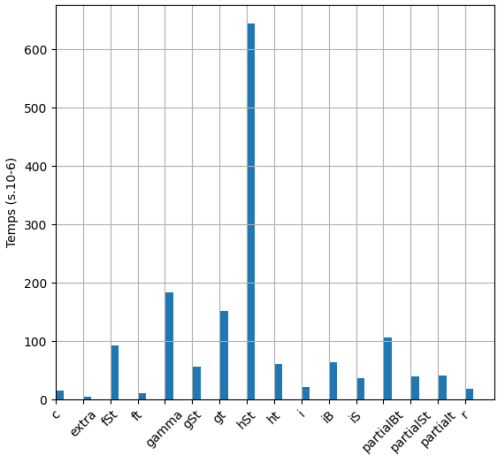
FIGURE 2.51 – Résultats en temps pour la couture de deux $(3, e)$-grilles le long d'une $(2, e)$-grille. Le g dans le nom des cas indique que les générateurs d'homologie peuvent être calculés (toutes les matrices sont maintenues). (a,c) $e = 25$, soit $265\ 302$ cellules à $t = 0$. (b,d) $e = 110$, soit $21\ 587\ 722$ cellules à $t = 0$.



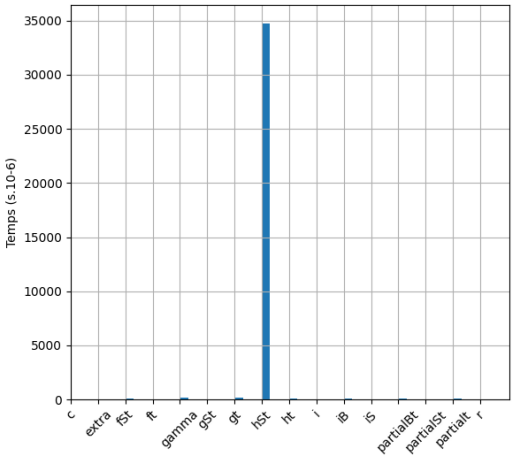
(a) Evolution du temps au fil des étapes $(k4e10n3g)$



(b) Evolution du temps au fil des étapes $(k4e35n3g)$

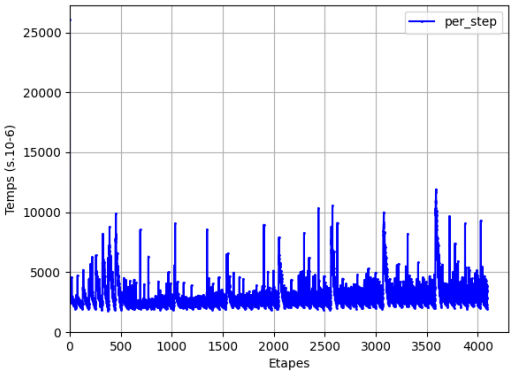


(c) Partage du temps entre les matrices $(k4e10n3g)$

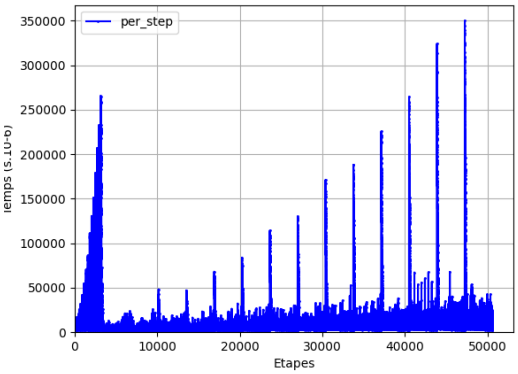


(d) Partage du temps entre les matrices $(k4e35n3g)$

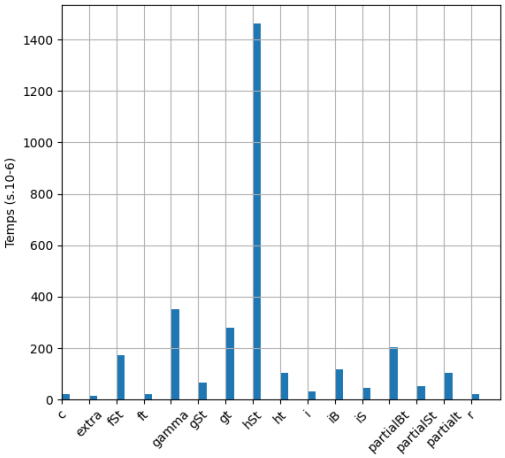
FIGURE 2.52 – Résultats en temps pour la couture de deux $(4, e)$-grilles le long d'une $(3, e)$-grille. Le g dans le nom des cas indique que les générateurs d'homologie peuvent être calculés (toutes les matrices sont maintenues). (a,c) $e = 10$ soit $388\,962$ cellules à $t = 0$. (b,d) $e = 35$ soit $50\,823\,362$ cellules à $t = 0$.



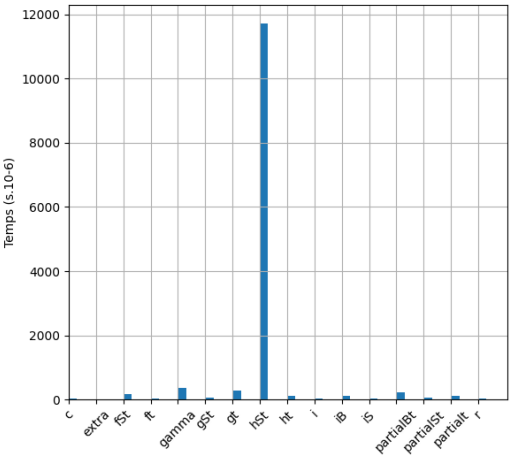
(a) Evolution du temps au fil des étapes ($k5e8n4g$)



(b) Evolution du temps au fil des étapes ($k5e15n4g$)



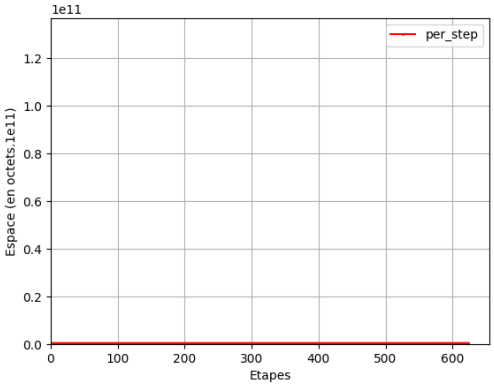
(c) Partage du temps entre les matrices ($k5e8n4g$)



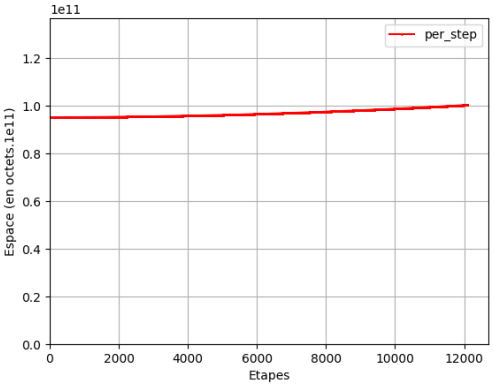
(d) Partage du temps entre les matrices ($k5e15n4g$)

FIGURE 2.53 – Résultats en temps pour la couture de deux $(5, e)$-grilles le long d'une $(4, e)$-grille. Le g dans le nom des cas indique que les générateurs d'homologie peuvent être calculés (toutes les matrices sont maintenues). (a,c) $e = 8$ soit $2\,839\,714$ cellules à $t = 0$. (b,d) $e = 15$ soit $57\,258\,302$ cellules à $t = 0$.

Espace. Les résultats sont exprimés sur une échelle de 0 à $137Go$. Les mesures sont illustrées sur les figures 2.54, 2.55 et 2.56. Observons que l'utilisation de la mémoire augmente cette fois-ci de manière visible. Pour autant, la tendance reste la même que dans le cas sans générateurs : l'augmentation est nettement inférieure au coût en espace des données initiales, à l'étape $t = 0$. Cela traduit le fait que le coût en espace d'une étape t donnée est égal au coût de l'étape $t = 0$ auquel s'ajoute la somme des variations en espace des étapes précédentes.

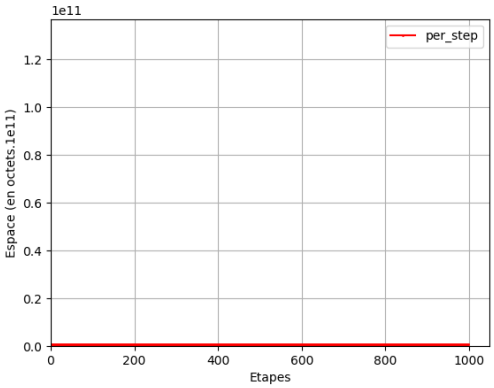


(a) Evolution de l'espace au fil des étapes $(k3e25n2g)$

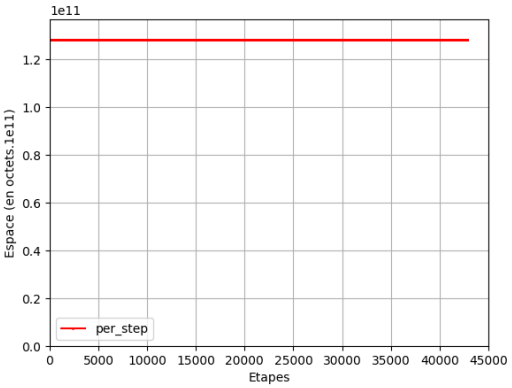


(b) Evolution de l'espace au fil des étapes $(k3e110n2g)$

FIGURE 2.54 – Résultats en espace pour la couture de deux $(3, e)$-grilles le long d'une $(2, e)$-grille. Le g dans le nom des cas indique que les générateurs d'homologie peuvent être calculés (toutes les matrices sont maintenues). (a) $e = 25$, soit $265\ 302$ cellules à $t = 0$. (b) $e = 110$, soit $21\ 587\ 722$ cellules à $t = 0$.

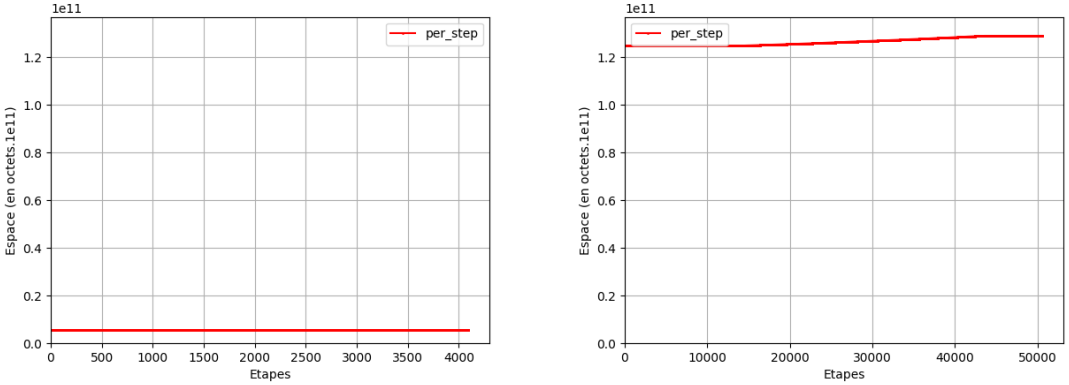


(a) Evolution de l'espace au fil des étapes $(k4e10n3g)$



(b) Evolution de l'espace au fil des étapes $(k4e35n3g)$

FIGURE 2.55 – Résultats en espace pour la couture de deux $(4, e)$-grilles le long d'une $(3, e)$-grille. Le g dans le nom des cas indique que les générateurs d'homologie peuvent être calculés (toutes les matrices sont maintenues). (a) $e = 10$ soit $388\ 962$ cellules à $t = 0$. (b) $e = 35$ soit $50\ 823\ 362$ cellules à $t = 0$.



(a) Evolution de l'espace au fil des étapes $(k5e8n4g)$

(b) Evolution de l'espace au fil des étapes $(k5e15n4g)$

FIGURE 2.56 – Résultats en espace pour la couture de deux $(5, e)$-grilles le long d'une $(4, e)$-grille. Le g dans le nom des cas indique que les générateurs d'homologie peuvent être calculés (toutes les matrices sont maintenues). (a) $e = 8$ soit $2\,839\,714$ cellules à $t = 0$. (b) $e = 15$ soit $57\,258\,302$ cellules à $t = 0$.

2.5 Conclusion et perspectives d'évolution

Conclusion. Nous utilisons le théorème des suites exactes courtes effectives (SECE) pour suivre, incrémentalement, l'homologie d'une structure topologique qui évolue dans un processus de construction par application d'identifications et/ou de désidentifications. Pour ce faire, on maintient une équivalence homologique γ^t à chaque étape t de la construction de la structure topologique. L'objectif est de calculer γ^{t+1}, l'équivalence homologique de l'étape $t+1$, par modification de γ^t. Dans ce chapitre, nous présentons une analyse du calcul des différents morphismes de γ^{t+1}. Ces derniers sont représentés sous forme de matrice. Pour chaque matrice, nous mettons en évidence des prérequis d'implémentation qui doivent être satisfaits pour optimiser le calcul de γ^{t+1}, c'est-à-dire en réutilisant au maximum les données de γ^t. À l'issue de l'analyse, nous constatons que l'implémentation repose essentiellement sur la structure de donnée utilisée pour représenter les matrices, creuses, et de la complexité des opérations qui l'accompagne. Nous réalisons une veille technique de quelques bibliothèques logicielles de matrices creuses. Il en résulte qu'aucune bibliothèque ne peut satisfaire l'ensemble des prérequis d'implémentation, notamment parce qu'elles sont basées sur des formats standards qui ne satisfont pas les prérequis. Pour cette raison, nous avons conceptualisé notre représentation matricielle. Elle fonctionne de pair avec les répertoires de cellules, qui permettent de suivre l'évolution des cellules d'une structure topologique dans un processus de construction. Nous proposons une implémentation du théorème SECE basée sur nos structures de données. Les résultats expérimentaux obtenus avec cette implémentation confirment les résultats théoriques, et montrent que :

— lorsque les générateurs d'homologie ne sont pas calculés, le temps de calcul de γ^{t+1} est fonction du nombre de cellules impactées par l'opération de passage de t à $t+1$, et de leur étoile directe. Le coût en espace de γ^t est égal au coût de $\gamma^{t=0}$ auquel s'ajoute la somme des variations en espace des étapes précédentes. La variation en espace du passage de t à $t+1$ est de l'ordre du nombre de cellules impactées par l'opération ;

— lorsque les générateurs d'homologie sont calculés, le temps de calcul de γ^{t+1} est lié au nombre de cellules de la structure topologique. Plus précisément, il dépend de la réduction $\rho^{S,t}$ et de la matrice $h^{S,t}$. Le coût en espace est calculé de la même façon que sans générateurs, mais la variation entre deux étapes est plus importante à une constante près.

Perspectives d'évolution. Dans ce chapitre, nous nous sommes essentiellement concentrés sur l'optimisation du temps de calcul de γ^{t+1} et sur l'évolution du comportement de ce calcul dans un processus de construction.

Optimiser l'espace. En particulier, l'implémentation que nous proposons est une preuve de concept qui nous permet d'étudier le comportement du théorème SECE en pratique au regard de ce à quoi nous nous attendions en théorie. Pour autant, nous pensons qu'il reste un travail à fournir sur l'amélioration de son utilisation de l'espace : malgré une évolution du coût en espace conforme à nos attentes, les valeurs mesurées restent élevées et peuvent être améliorées. Une des pistes évoquées pour cela est d'étendre la représentation implicite de données aux complexes de chaînes, dont certains peuvent être vus comme une variation des complexes de chaînes initiaux du processus de construction étudié : à $t=0$, $(C^t, \partial^t) = (C^{B,t}, \partial^{B,t})$, il doit donc être possible d'éviter de représenter explicitement l'entièreté de ces deux complexes de chaînes, contrairement à ce que nous faisons.

Étendre l'étude expérimentale. Une autre piste d'amélioration des travaux est l'extension de l'étude expérimentale à l'opération de désidentification. En effet, nous avons vu que l'application du théorème SECE à l'identification et à la désidentification repose sur quasiment les mêmes prérequis d'implémentation, avec un changement mineur sur la fusion de matrice où la "grosse" matrice n'est plus en bas à droite mais en haut à gauche. Étudier expérimentalement la désidentification permettrait de confirmer en pratique cette observation.

Étudier en profondeur la notion de réduction. Lorsque les générateurs d'homologie sont calculés, le temps de calcul dépend, au pire des cas, du nombre de cellules de la structure topologique [16]. Cela est dû à la réduction $\rho^{S,t}$ de l'équivalence homologique maintenue, et plus particulièrement au morphisme $h^{S,t}$ qui présente deux particularités :

— il a pour domaine et pour codomaine le complexe de chaînes intermédiaire $(C^{B,t}, \partial^{B,t})$;

— en toute généralité, pour une p-cellule donnée, la taille de la p-chaîne obtenue par $h^{S,t}$ est au plus le nombre de p-cellules de $C^{B,t}$.

Or, le complexe de chaînes intermédiaire $(C^{B,t}, \partial^{B,t})$ peut être réduit de différentes façons et, selon celle choisie, la taille maximale des chaînes obtenues par $h^{S,t}$ diffère. Plus cette taille est petite, plus le théorème SECE est performant. Étudier en profondeur la notion de réduction permettrait d'établir l'existence d'une réduction optimale limitant la taille de la plus grande chaîne obtenue par $h^{S,t}$, et donc d'améliorer les performances du théorème SECE.

16. Lorsque les générateurs d'homologie sont calculés, le temps de calcul dépend en réalité de la dimension des cellules impliquées dans l'opération de passage de t à $t+1$ et de la réduction $\rho^{S,t}$. En particulier, si l'on suppose qu'une seule p-cellule est impliquée, le temps de calcul dépend du nombre de p-cellules du complexe de chaînes intermédiaire de l'équivalence homologique maintenue. Dans certains cas particuliers, cela peut représenter la totalité des cellules de la structure topologique, il s'agit donc d'une fourchette haute. Cela est discuté plus en détail dans le cas des grilles en annexe D.2

Chapitre 3

Calcul incrémental d'un objet distribué

Contenu du chapitre

3.1 Préambule

3.1.1 Contexte et terminologie

3.1.1-a Contexte.

Pour rappel, le contexte général des travaux présentés dans cette thèse est le calcul incrémental de l'homologie d'un objet qui évolue dans un processus de construction par application d'une séquence d'identifications et/ou de désidentifications. Dans ce chapitre, on suppose que cet objet est trop complexe pour être manipulé par une seule *unité de calcul*. Il est donc éclaté en plusieurs morceaux, distribués sur plusieurs unités de calcul. L'objet distribué est donc représenté implicitement au travers de sa *distribution* et d'une *identification de reconstruction*.

Exemple. La figure 3.2a illustre la distribution et l'identification de reconstruction d'un objet à l'étape t d'un processus de construction.

Lorsqu'il subit une opération, le faisant passer d'une étape t à une étape $t+1$, celle-ci est *répartie* entre la distribution et l'identification de reconstruction afin de les faire correspondre à l'objet distribué de l'étape $t+1$. À ce jour, nous nous sommes intéressés à la répartition d'une identification, étudiée dans la section 3.2, et la répartition d'une désidentification reste à traiter.

Dans la section 3.3, nous présentons nos travaux concernant l'implémentation de la répartition d'une identification sur laquelle nous avons travaillé avec Guillaume DAMIAND, du laboratoire LIRIS. En effet, notre point de départ pour l'implémentation est le code lié aux travaux présentés dans [88]. On y trouve notamment la notion de carte orientée distribuée, dont nous nous servons pour distribuer l'objet. Nous avons fragmenté l'implémentation en 3 grandes étapes :

1. lier le code de [88] et le logiciel développé dans le cadre du Chapitre 2, en séquentiel, avec une seule unité de calcul. Plus précisément, cela consiste à calculer l'équivalence homologique induite par une carte orientée (cf. paragraphe 1.2.3-c), et à construire les identifications induites par les opérations de couture [1] appliquées à cette dernière ;
2. étendre cela à une carte orientée distribuée, avec plusieurs unités de calcul. Les coutures sont définies à partir de la distribution, c'est-à-dire que les identifications impliquent soit des cellules appartenant à une même unité de calcul, soit des cellules appartenant à des unités de calcul différentes ;
3. enfin, définir les opérations de couture non plus à partir de la distribution mais à partir de la carte orientée distribuée, c'est-à-dire que les cellules impliquées dans les identifications appartiennent à des unités de calcul quelconques. Dans ce cas, des calculs sont nécessaires pour déterminer la partie des identifications qui implique de modifier la distribution, et celle qui implique de modifier l'identification de reconstruction.

À ce jour, l'étape 2 est implémentée et reste à tester, l'étape 3 reste à implémenter.

3.1.1-b Terminologie.

Distribution et identification de reconstruction. Distribuer un objet désigne l'action de l'éclater en m morceaux M_l et d'attribuer une unité de calcul u_l à chacun d'eux, pour l compris entre 0 et $m-1$ inclus. La distribution de l'objet est l'ensemble des couples (M_l, u_l) ainsi formés. L'identification de reconstruction désigne l'identification à appliquer aux morceaux de la

1. Nous utilisons le terme "couture" pour distinguer l'opération appliquée à la carte généralisée et l'identification qu'elle induit sur l'ensemble semi-simplicial numéroté associé à la carte (cf. paragraphe 1.2.3-c).

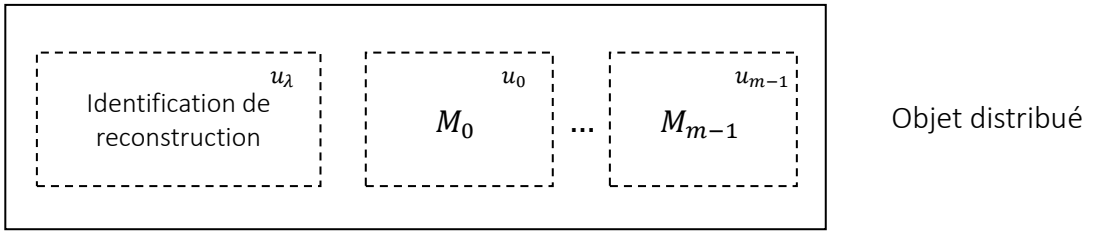


FIGURE 3.1 – Un objet éclaté en m morceaux M_l, distribués sur m unités de calcul u_l, pour l compris entre 0 et $m-1$ inclus. L'unité de calcul critique u_λ contient l'identification de reconstruction.

distribution pour reconstruire l'objet distribué. Plus simplement, nous dirons que l'identification de reconstruction est appliquée à la distribution.

Unités de calcul. Le terme "unité de calcul" désigne une entité capable de contenir des données et d'effectuer des calculs à partir de ses données. Supposons que l'objet est distribué sur m unités de calcul, et nommons u_l la l-ème unité de calcul, pour l compris entre 0 et $m-1$ inclus. Elle contient :

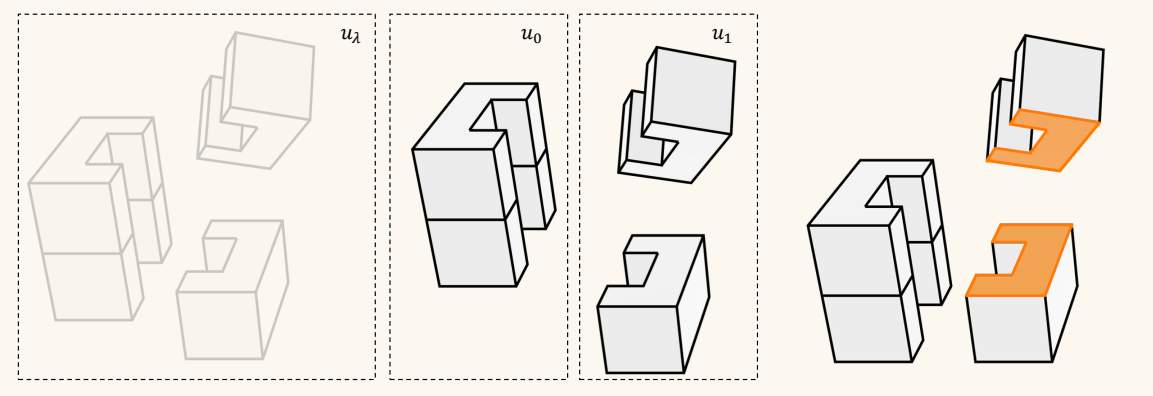
— le morceau M_l de l'objet distribué, représenté par une structure topologique ;

— l'équivalence homologique γ_l^t, contenant notamment le complexe de chaînes (C_l^t, ∂_l^t) induit par M_l.

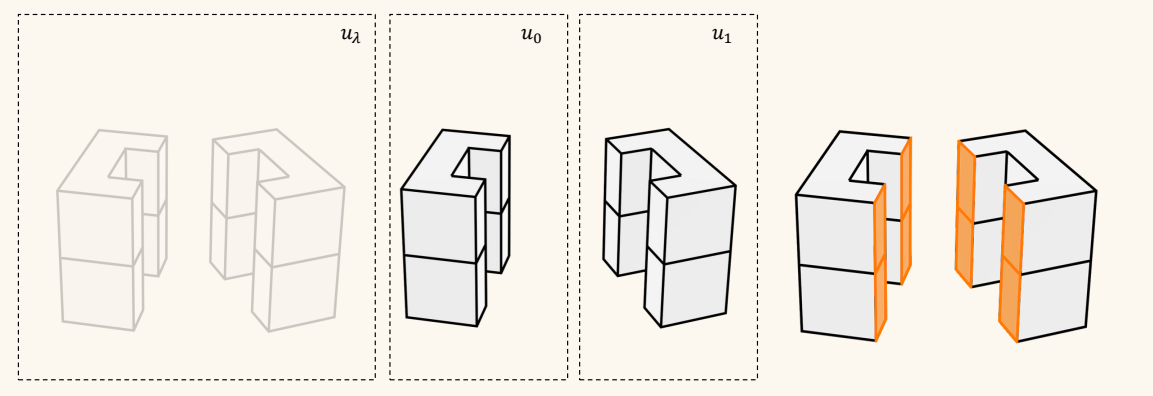
On distingue une unité de calcul particulière, u_λ, qui contient l'identification de reconstruction. Au total, la distribution d'un objet implique donc $m+1$ unités de calcul, comme l'illustre la figure 3.1.

Remarque. Par souci de clarté, nous supposons ici que u_λ est une unité de calcul à part entière. Cependant, elle peut être choisie parmi les autres unités de calcul, auquel cas la distribution implique au total m unités de calcul.

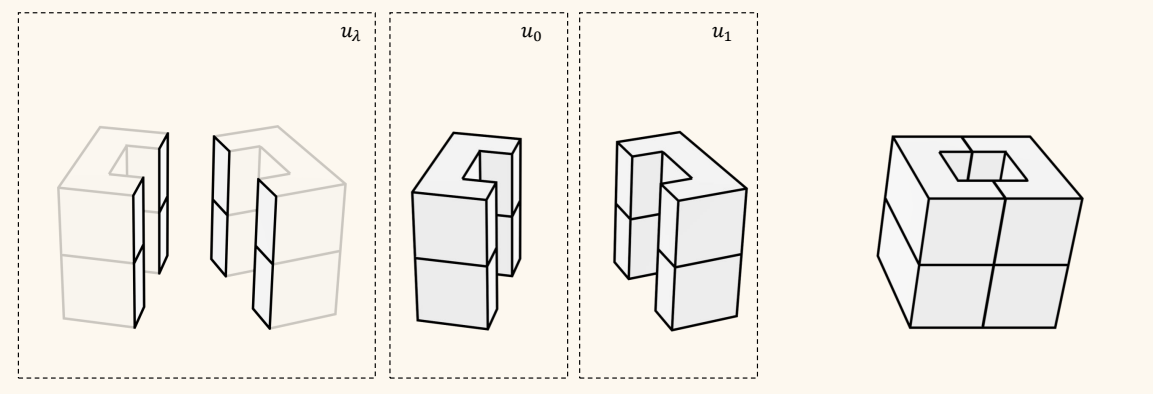
Répartition d'une opération. Lorsqu'une opération, en l'occurrence une identification, est appliquée à l'objet distribué pour le faire passer d'une étape t à une étape $t+1$, sa distribution et l'identification de reconstruction ne lui correspondent plus. Répartir une telle identification veut dire que l'on modifie la distribution et l'identification de reconstruction pour les faire correspondre à l'objet distribué de l'étape $t+1$. La figure 3.2 illustre la répartition de deux identifications.



(a) Étape t.



(b) Étape $t+1$.



(c) Étape $t+2$.

FIGURE 3.2 – La distribution d'un objet sur 2 unités de calcul u_0 et u_1, évoluant dans un processus de construction en 3 étapes, le passage d'une étape à l'autre se fait en identifiant les cellules en orange entre elles. L'identification de reconstruction est illustrée sur u_λ, et consiste à identifier les cellules qui ne sont pas transparentes. Enfin, notons que le passage de t à $t+1$ induit une modification de M_1, et que le passage de $t+1$ à $t+2$ alimente l'identification de reconstruction.

3.1.2 Objectifs et hypothèses

Le premier objectif des travaux présentés dans ce chapitre est de calculer incrémentalement l'homologie de l'objet distribué. Pour ce faire, on applique le théorème SECE à l'identification [2] de reconstruction. En particulier :

— il résulte de ce calcul une équivalence homologique γ^t, associée à l'objet distribué de l'étape t. Or, cet objet est supposé trop complexe pour être manipulé par une seule unité de calcul, il en est donc de même à fortiori pour γ^t ;

— contrairement à l'usage que nous faisons jusque-là du théorème SECE, on ne souhaite pas maintenir γ^t mais simplement utiliser son "petit" complexe de chaînes $(C^{S,t}, \partial^{S,t})$ en entrée d'une méthode de calcul d'homologie (FNS par exemple, cf. sous-section 1.4.1). On suppose que ce complexe de chaînes est suffisamment petit pour être représenté sur une unité de calcul ; il représente la seule donnée de γ^t explicitement représentée pour calculer l'homologie de l'objet distribué. On ne calcule donc pas la totalité de l'équivalence homologique γ^t. Notons que, pour l'instant, cela implique qu'il n'y a pas de correspondance directe entre les générateurs du petit complexe de chaînes et ceux des autres complexes de chaînes de γ^t. La levée de cette limitation reste à étudier. La figure 3.3 illustre un exemple d'application du théorème SECE à une identification de reconstruction ;

Il découle de l'analyse de la section 2.1 que plus l'identification de reconstruction implique de cellules, plus l'application du théorème SECE est coûteuse. Cela nous mène au second objectif des travaux : minimiser l'identification de reconstruction à chaque étape. Dit autrement, lors d'une répartition d'identifications, on cherche d'abord à modifier la distribution avant de modifier l'identification de reconstruction. En particulier, cela implique que si deux cellules sont identifiées dans l'identification de reconstruction, alors elles appartiennent forcément à des unités de calcul différentes. Les hypothèses du chapitre sont répertoriées ci-dessous.

Hypothèses.

(i) L'objet et l'équivalence homologique γ^t qui lui est associée sont trop complexes pour être manipulés par une seule unité de calcul. L'objet est éclaté en plusieurs morceaux distribués sur plusieurs unités de calcul, chacune contenant une équivalence homologique γ^t_l ;

(ii) Le petit complexe de chaînes $(C^{S,t}, \partial^{S,t})$ est suffisamment petit pour être représenté sur une unité de calcul ;

(iii) Si deux cellules sont identifiées dans la reconstruction, alors elles appartiennent forcément à des unités de calcul distinctes.

2. Pour rappel, l'application du théorème SECE à une identification est présentée dans la sous-section 1.5.1.

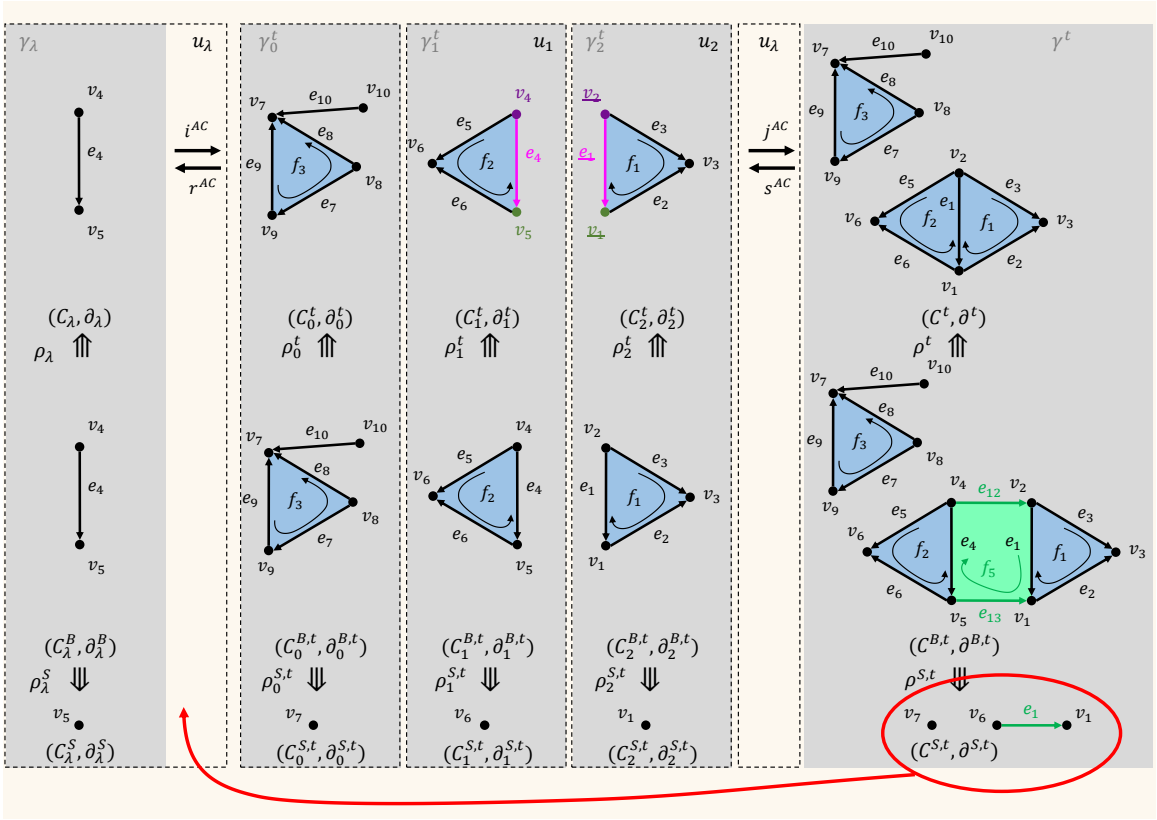


FIGURE 3.3 – Application du théorème SECE à une identification de reconstruction pour un objet distribué sur 3 unités de calcul u_0, u_1 et u_2. Les cellules identifiées entre elles dans la distribution sont mises en évidence par une même couleur. Les cellules survivantes sont soulignées. Les équivalences homologiques sont mises en évidence par un fond gris. En vert fluo, les cellules ajoutées au "gros" et au "petit" complexe de chaînes de γ^t par rapport à ceux de la distribution. Seul le petit complexe de chaînes de γ^t, encerclé en rouge, est calculé sur u_λ. Pour cela, on récupère les "petits" complexes de chaînes de γ_0^t, γ_1^t et γ_2^t sur u_λ, ainsi que l'image des cellules en couleur dans la distribution dans ces petits complexes de chaînes (via les réductions ρ_1^t, $\rho_1^{S,t}$ et ρ_2^t, $\rho_2^{S,t}$).

3.2 Répartition d'une identification

Dans cette section, nous présentons un algorithme de répartition d'une identification permettant de calculer la distribution et l'identification de reconstruction correspondant à un objet distribué après qu'une identification lui soit appliquée. Au total, cet algorithme implique 4 identifications illustrées sur la figure 3.4 :

— l'identification de reconstruction de l'objet distribué de l'étape t ;

— l'identification appliquée à l'objet distribué pour le faire passer de l'étape t à l'étape $t+1$;

— l'identification appliquée à la distribution de l'étape t pour obtenir celle de l'étape $t+1$, calculée par l'algorithme de répartition ;

— l'identification de reconstruction de l'objet distribué de l'étape $t+1$, calculée par l'algorithme de répartition.

Afin d'éviter une confusion entre objet distribué et distribution des étapes t et $t+1$, nous nommons :

— A la distribution de l'étape t ;

— B la distribution de l'étape $t+1$;

— C l'objet distribué de l'étape t ;

— D l'objet distribué de l'étape $t+1$.

Ces notations sont illustrées sur la figure 3.5. Dans un premier temps, nous présentons l'algorithme de répartition d'une identification sur les ensembles semi-simpliciaux [3]. Dans un second temps, nous étudions ses implications sur les suites exactes courtes effectives (SECE) induites par les 4 identifications.

3. L'algorithme de répartition d'une identification s'étend à tout type de structure topologique sur laquelle est définie l'identification et qui induit un complexe de chaînes, sous réserve que ses cellules respectent certaines propriétés (cf. paragraphe 1.3.2-d).

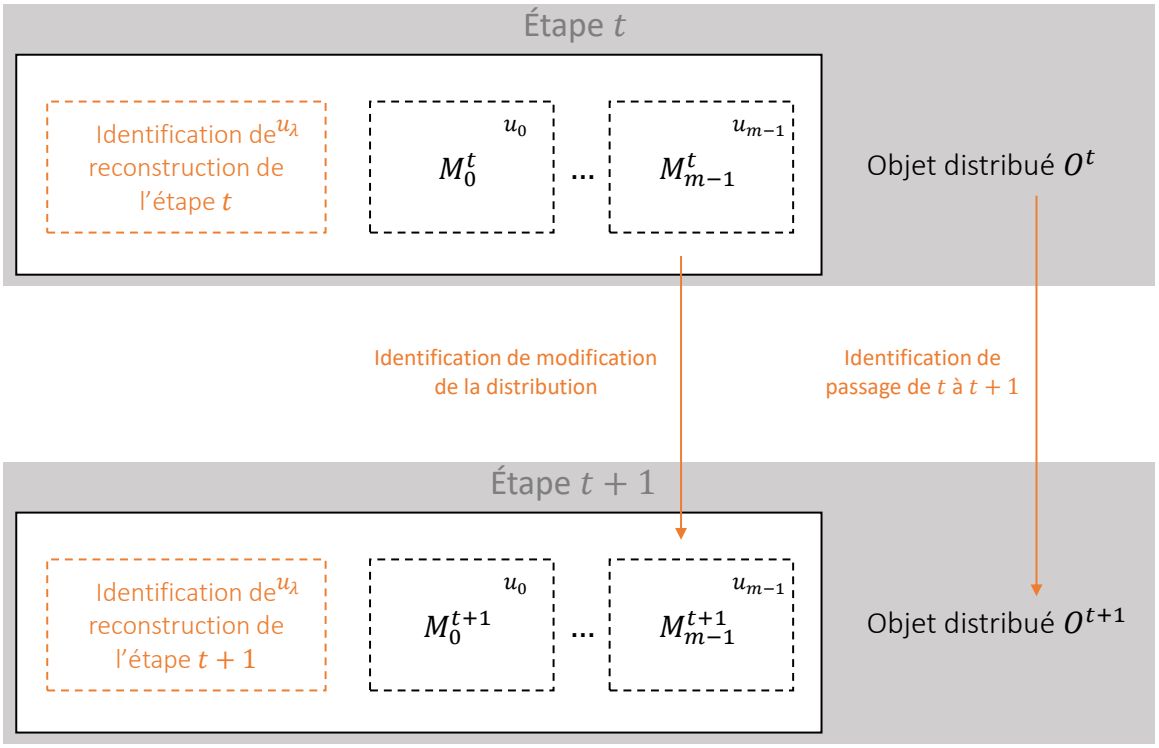


FIGURE 3.4 – Les 4 identifications impliquées dans la répartition d'une identification, mises en évidence en orange.

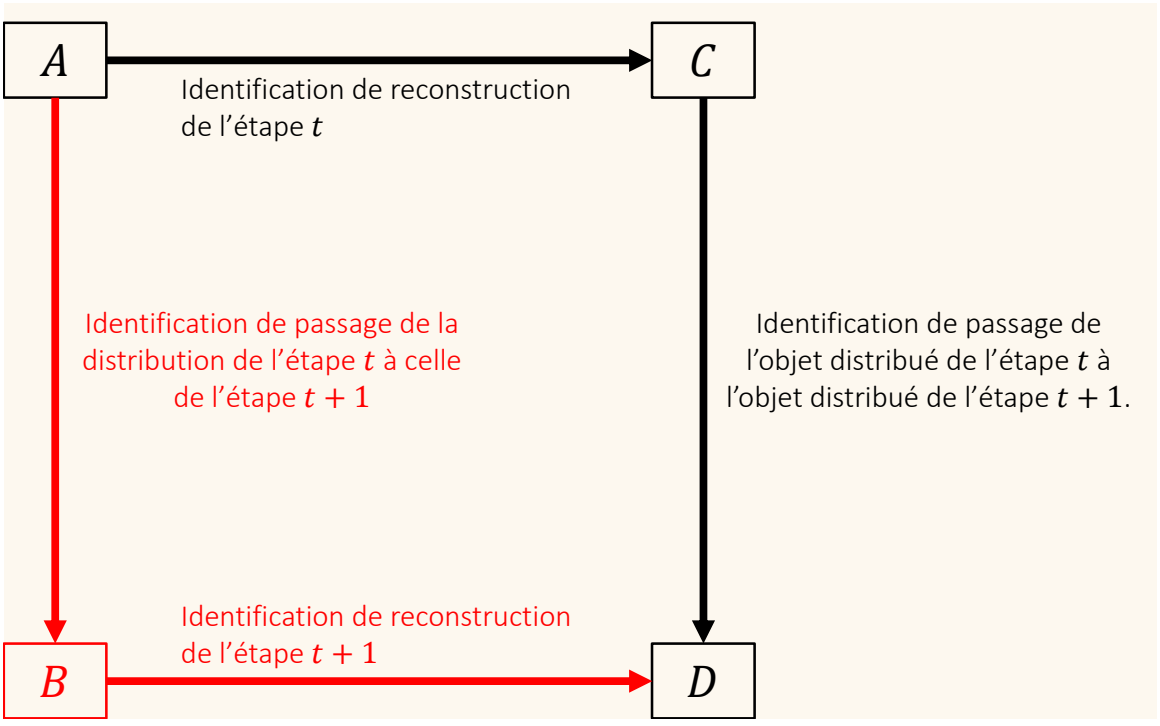


FIGURE 3.5 – La convention de nommage que nous utilisons. En rouge, les éléments calculés par l'algorithme.

3.2.1 Répartition sur les ensembles semi-simpliciaux

Pour désigner plus simplement les différentes identifications impliquées dans une répartition, nous notons (I^{XY}, ζ^{XY}) l'identification appliquée à l'ensemble semi-simplicial [4] (K^X, d^X) (ou $\oplus_{l=0}^{m-1}(K_l^X, d_l^X)$) et de laquelle résulte l'ensemble semi-simplicial (K^Y, d^Y). Les éléments manipulés au cours de la répartition d'une identification appliquée à un ensemble semi-simplicial sont illustrés sur la figure 3.6. Nous présentons le principe de l'algorithme et son fonctionnement, que nous illustrons ensuite sur un exemple détaillé.

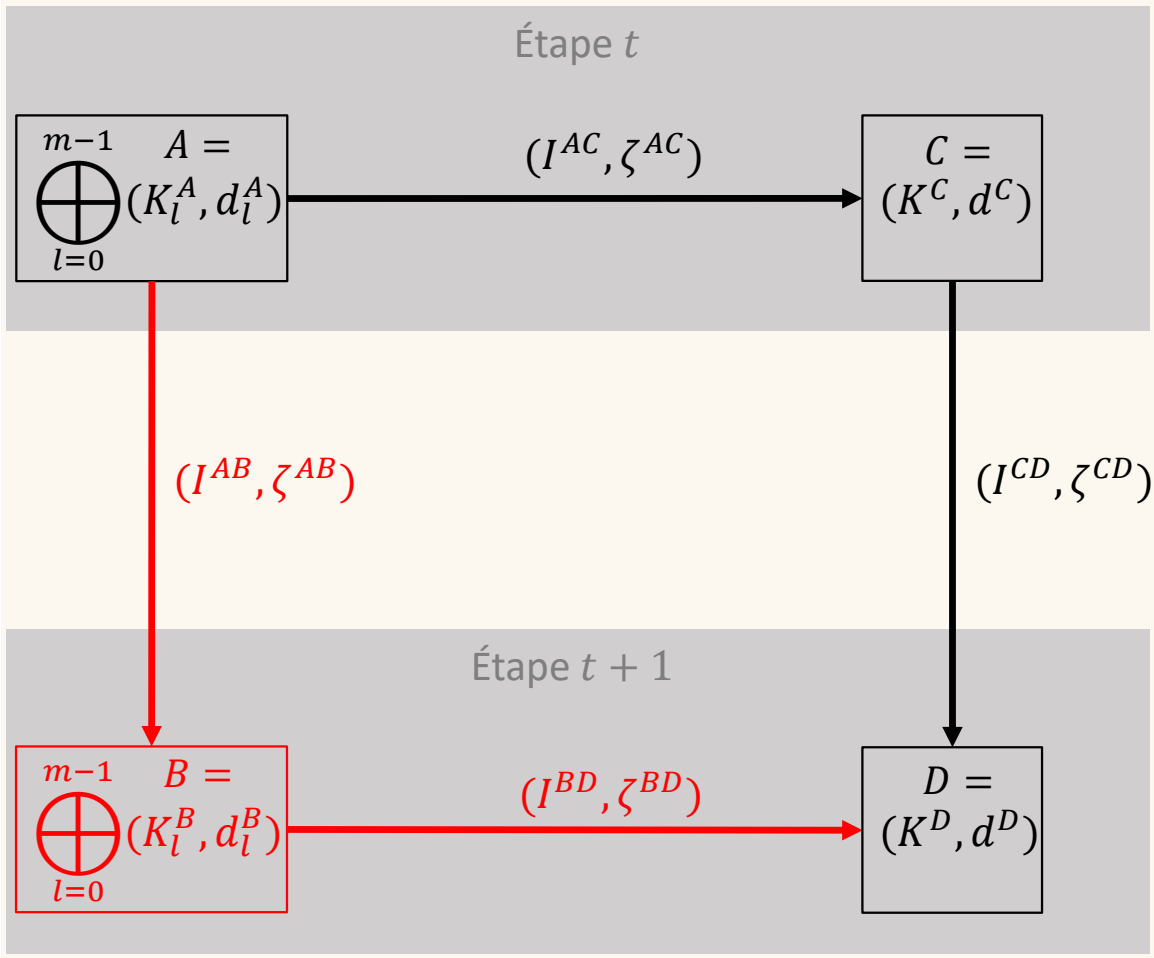


FIGURE 3.6 – Vue d'ensemble des éléments impliqués dans la répartition de l'identification (I^{CD}, ζ^{CD}). En rouge, les éléments calculés par l'algorithme.

4. Pour rappel, la définition de l'identification pour des ensembles semi-simpliciaux est présentée dans le paragraphe 1.2.1-b.

3.2.1-a Principe

Pour illustrer le principe de l'algorithme de répartition, nous nous appuyons sur l'exemple illustré sur la figure 3.7. Le principe s'articule autour de deux points clefs :

— la composition de (I^{AC}, ζ^{AC}) et (I^{CD}, ζ^{CD}) ;

— la restriction de cette composition par unités de calcul.

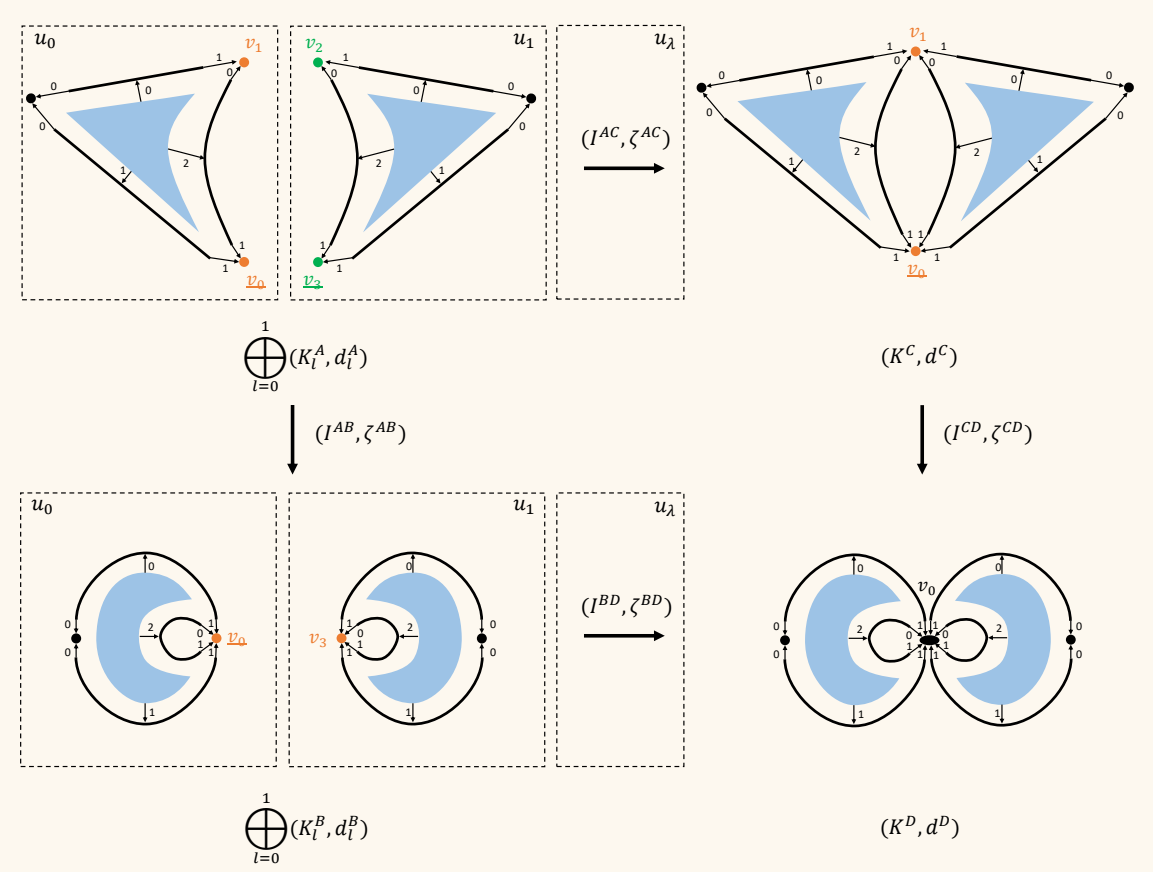


FIGURE 3.7 – La répartition d'une identification (I^{CD}, ζ^{CD}), dans laquelle les sommets v_0 et v_1 sont identifiés entre eux. Les simplexes identifiés dans (I^{CD}, ζ^{CD}), (I^{AB}, ζ^{AB}) et (I^{BD}, ζ^{BD}) sont mis en évidence en couleur sur (K^C, d^C), $\oplus_{l=0}^{1}(K_l^A, d_l^A)$ et $\oplus_{l=0}^{1}(K_l^B, d_l^B)$. Les simplexes survivants sont surlignés. Pour une identification donnée, les simplexes d'une même classe d'équivalence sont de même couleur. L'identification (I^{AC}, ζ^{AC}) n'est pas mise en évidence par souci de clarté. Elle consiste à identifier v_1 avec v_2 et v_0 avec v_3, v_0 et v_1 sont survivants.

Composition de (I^{AC}, ζ^{AC}) et (I^{CD}, ζ^{CD}). L'ensemble semi-simplicial (K^C, d^C) est représenté implicitement au travers de sa distribution $\oplus_{l=0}^{m-1}(K_l^A, d_l^A)$ et de l'identification de reconstruction (I^{AC}, ζ^{AC}). Dit autrement, les simplexes de (K^C, d^C) sont tous des simplexes survivants de (I^{AC}, ζ^{AC}), et sont donc "représentés" par plusieurs simplexes dans la distribution $\oplus_{l=0}^{m-1}(K_l^A, d_l^A)$.

Notons $[\sigma]^{CD}$ la classe d'équivalence d'un simplexe σ dans I^{CD} et $[\sigma]^{AC}$ sa classe d'équivalence dans I^{AC}. Lorsqu'on identifie deux simplexes σ et σ' de (K^C, d^C) en choisissant σ comme

survivant, ce dernier est le seul des deux que l'on retrouve dans (K^D, d^D). Cela veut dire que parmi l'ensemble des simplexes de $[\sigma]^{AC} \cup [\sigma']^{AC}$, seul σ se retrouve dans (K^D, d^D). Dit autrement, à l'exception de σ, aucun des simplexes de $[\sigma]^{AC} \cup [\sigma']^{AC}$ ne survit jusque dans (K^D, d^D). Ils doivent donc tous être identifiés dans (I^{AB}, ζ^{AB}) et (I^{BD}, ζ^{BD}) ; en toute généralité, le choix de la répartition des simplexes entre ces 2 identifications est libre. Du fait de l'hypothèse (iii), nous utilisons une restriction par unité de calcul pour déterminer cette répartition, comme nous le verrons dans le paragraphe suivant. L'ensemble $[\sigma]^{AC} \cup [\sigma']^{AC}$ est la *composition* de $[\sigma]^{AC}$ induite par $[\sigma]^{CD} = \{\sigma, \sigma'\}$. Plus généralement, la composition induite par une classe $[\sigma']^{CD}$ de I^{CD} est l'ensemble défini par :

$$\bigcup_{\sigma \in [\sigma']^{CD}} [\sigma]^{AC}$$

Exemple. La figure 3.7 illustre la répartition d'une identification (I^{CD}, ζ^{CD}) dans laquelle sont identifiés deux sommets v_0 et v_1, et où v_0 est survivant. Observons que :

- les simplexes v_0 et v_1 de (K^C, d^C) sont tout deux survivants dans (I^{AC}, ζ^{AC}) ;
- ils sont "représentés" respectivement par v_0 et v_3 et v_1 et v_2 dans $\oplus_{l=0}^{m-1}(K_l^A, d_l^A)$.

L'ensemble $[v_0]^{AC} \cup [v_1]^{AC} = \{v_0, v_1, v_2, v_3\}$ est la composition de $[v_0]^{AC} = \{v_0, v_3\}$, $[v_1]^{AC} = \{v_1, v_2\}$ induite par $[v_0]^{CD} = \{v_0, v_1\}$. Seul le simplexe v_0 se retrouve dans (K^D, d^D) ; les simplexes v_1, v_2 et v_3 doivent donc être identifiés soit dans (I^{AB}, ζ^{AB}), soit dans (I^{BD}, ζ^{BD}).

Restriction par unités de calcul. En toute généralité, la seule condition sur la répartition des simplexes de la composition de (I^{AC}, ζ^{AC}) et (I^{CD}, ζ^{CD}) est que les survivants de (I^{CD}, ζ^{CD}) soient survivants dans (I^{AB}, ζ^{AB}) et (I^{BD}, ζ^{BD}). Pour autant, rappelons que l'un de nos objectifs est de minimiser l'identification de reconstruction à chaque étape, et que cela induit l'hypothèse (iii). La répartition que nous appliquons à la composition de (I^{AC}, ζ^{AC}) et (I^{CD}, ζ^{CD}) est une restriction par unités de calcul, c'est-à-dire que l'on cherche dans la composition les simplexes appartenant à une même unité de calcul. Ces derniers sont identifiés entre eux dans (I^{AB}, ζ^{AB}), et "le reste" est identifié dans (I^{BD}, ζ^{BD}). Nous choisissons [5] de définir les simplexes survivants de (I^{AB}, ζ^{AB}) et (I^{BD}, ζ^{BD}) en fonction de (I^{CD}, ζ^{CD}) : tous les simplexes d'une classe d'équivalence $[\sigma]^{CD}$ de I^{CD} sont survivants dans (I^{AB}, ζ^{AB}), de même que les simplexes de $[\sigma]^{AC}$ si σ est le simplexe survivant de $[\sigma]^{CD}$, qui est aussi survivant dans (I^{BD}, ζ^{BD}).

Exemple. Sur la figure 3.7, la restriction de $[v_0]^{AC} \cup [v_1]^{AC} = \{v_0, v_1, v_2, v_3\}$ par unités de calcul est telle que :

- v_0 et v_1 sont identifiés dans (I^{AB}, ζ^{AB}) car ils appartiennent tout deux à u_0. En particulier, v_0 est survivant car il est le simplexe que l'on doit retrouver dans (K^D, d^D) (de fait, il appartient à $[v_0]^{AC}$) ;
- v_2 et v_3 sont identifiés dans (I^{AB}, ζ^{AB}) car ils appartiennent tout deux à u_1. En particulier, v_3 est survivant car il appartient à $[v_0]^{AC}$;

Le "reste", c'est-à-dire $\{v_0, v_3\}$ est identifié dans (I^{BD}, ζ^{BD}) et v_0 est survivant.

5. Comme nous le verrons dans le paragraphe 3.2.2-b, ce choix permet d'*optimiser* certains calculs.

3.2.1-b Fonctionnement

Nous présentons en détail le fonctionnement de l'algorithme de répartition d'une identification (I^{CD}, ζ^{CD}) entre (I^{AB}, ζ^{AB}) et (I^{BD}, ζ^{BD}), en précisant les données qu'il prend en entrée et les calculs qu'il réalise. Les exemples que nous donnons au fil de ce paragraphe s'appuient sur la figure 3.7.

Données de départ. L'ensemble semi-simplicial (K^C, d^C) est une donnée "virtuelle". Il est représenté au travers de sa distribution $\oplus_{l=0}^{m-1}(K_l^A, d_l^A)$ sur m unités de calcul et de l'identification de reconstruction (I^{AC}, ζ^{AC}). Comme nous l'avons vu, les données de départ de la répartition sont :

— l'identification (I^{CD}, ζ^{CD}) ;

— la distribution $\oplus_{l=0}^{m-1}(K_l^A, d_l^A)$ telle que (K_l^A, d_l^A) est un ensemble semi-simplicial appartenant à l'unité de calcul u_l pour l compris entre 0 et $m-1$;

— l'identification de reconstruction (I^{AC}, ζ^{AC}) appliquée à $\oplus_{l=0}^{m-1}(K_l^A, d_l^A)$. Du fait de l'hypothèse (iii), on suppose que si deux cellules sont identifiées dans (I^{AC}, ζ^{AC}), alors elles appartiennent à des unités de calcul distinctes.

Exemple. (I^{AC}, ζ^{AC}) est caractérisée[6] par $I^{AC} = \{\{v_0, v_3\}, \{v_1, v_2\}\}$. Cela implique que :

— v_0 et v_3 appartiennent à des unités de calcul distinctes, en l'occurrence u_0 et u_1 ;

— v_1 et v_2 appartiennent à des unités de calcul distinctes, en l'occurrence u_0 et u_1.

Calculs. La répartition d'une identification consiste à calculer (I^{AB}, ζ^{AB}) et (I^{BD}, ζ^{BD}). Nous avons vu que (I^{AB}, ζ^{AB}) peut être décomposée en plusieurs identifications (I_l^{AB}, ζ^{AB}), chacune d'entre elles étant propre à une unité de calcul u_l.

Exemple. (I^{AB}, ζ^{AB}) est caractérisée par $I^{AB} = \{\{v_0, v_1\}, \{v_3, v_2\}\}$. Elle peut être décomposée en deux identifications (I_0^{AB}, ζ_0^{AB}), propre à u_0, et (I_1^{AB}, ζ_1^{AB}), propre à u_1. Elles sont caractérisées par $I_0^{AB} = \{v_0, v_1\}$ et $I_1^{AB} = \{v_3, v_2\}$.

L'identification (I^{BD}, ζ^{BD}) représente l'identification de reconstruction de l'étape $t+1$. À ce titre, du fait de l'hypothèse (iii), on suppose que si deux cellules sont identifiées dans (I^{BD}, ζ^{BD}), alors elles appartiennent à des unités de calcul distinctes.

Exemple. (I^{BD}, ζ^{BD}) est caractérisée par $I^{BD} = \{\{v_0, v_3\}\}$ et $v_0\zeta^{BD} = v_3\zeta^{BD} = v_0$. Cela implique que v_0 et v_3 appartiennent à des unités de calcul distinctes, en l'occurrence u_0 et u_1.

Le calcul de ces deux identifications se fait en partant des classes d'équivalences de I^{CD}. Soit $[\sigma]^{CD} = \{\sigma_0, \ldots, \sigma_{q-1}\}$ une classe d'équivalence de I^{CD} dont σ_0 est le simplexe survivant. La répartition de $[\sigma]^{CD}$ implique 3 étapes de calcul :

1. d'abord, calculer l'union des classes d'équivalence dans I^{AC} de tous les simplexes appartenant à $[\sigma]^{CD}$. Notons $[\sigma]^{AC} = [\sigma_0]^{AC} \cup \ldots \cup [\sigma_{q-1}]^{AC}$ l'ensemble qui en résulte ;
2. ensuite, restreindre $[\sigma]^{AC}$ aux unités de calcul. Rappelons que d'après l'hypothèse (iii) et pour i compris entre 0 et $q-1$ inclus, les simplexes de $[\sigma_i]^{AC}$ appartiennent tous à des unités de calcul distinctes. Il existe donc au plus q simplexes de $[\sigma]^{AC}$ appartenant à une

6. Pour simplifier la notation, on ne représente que les classes d'équivalence contenant au moins deux simplexes. De plus, pour l'ensemble de ce chapitre, nous faisons le choix de ne plus distinguer la dimension des simplexes dans la notation.

même unité de calcul. Soit X_l l'ensemble des simplexes de $[\sigma]^{AC}$ appartenant à u_l. Ils sont identifiés entre eux dans (I^{AB}, ζ^{AB}), et le simplexe survivant σ_l^{AB} est,

— s'il existe, le simplexe de X_l tel que $\sigma_l^{AB}\zeta^{AC} = \sigma_0$;

— choisi arbitrairement parmi les simplexes de X_l sinon.

3. enfin, définir $X = (\cup_{l=0}^{m-1}(X_l \setminus \{\sigma_l^{AB}\}))$, où σ_l^{AB} désigne le simplexe survivant de X_l. Les simplexes de $[\sigma]^{AC} \setminus X$ sont identifiés entre eux dans (I^{BD}, ζ^{BD}). Le simplexe survivant est $\sigma = \sigma_0$. Notons que $[\sigma]^{AC} \setminus X$ représente le "reste" dont nous parlions dans le paragraphe 3.2.1-a.

Toutes les classes d'équivalence de (I^{AC}, ζ^{AC}) qui n'ont pas de rapport avec l'identification (I^{CD}, ζ^{CD}) se retrouvent en l'état dans (I^{BD}, ζ^{BD}), et $\sigma\zeta^{BD} = \sigma\zeta^{AC}$ pour les simplexes σ de ces classes.

Exemple. Sur la figure 3.7, l'identification (I^{CD}, ζ^{CD}) est caractérisée par $I^{CD} = \{\{v_0, v_1\}\}$ et $v_0\zeta^{CD} = v_1\zeta^{CD} = v_0$. Considérons $[v_0]^{CD} = \{v_0, v_1\}$. Sa répartition implique 3 étapes de calcul :

1. d'abord, le calcul de $[v_0]^{AC} \cup [v_1]^{AC} = \{v_0, v_1, v_2, v_3\}$;
2. ensuite, la restriction de $[v_0]^{AC} \cup [v_1]^{AC}$ aux unités de calcul. Observons que les simplexes de $[v_0]^{AC} = \{v_0, v_3\}$ et $[v_1]^{AC} = \{v_1, v_2\}$ appartiennent tous à des unités de calcul distinctes. Il existe donc au plus 2 simplexes de $[v_0]^{AC} \cup [v_1]^{AC}$ appartenant à une même unité de calcul, un dans $[v_0]^{AC}$ et l'autre dans $[v_1]^{AC}$. On obtient :
   — $X_0 = \{v_0, v_1\}$ pour u_0. Ces simplexes sont identifiés dans (I^{AB}, ζ^{AB}) et le survivant est v_0 car $v_0\zeta^{AC} = v_0$;
   — $X_1 = \{v_3, v_2\}$ pour u_1. Ces simplexes sont identifiés dans (I^{AB}, ζ^{AB}) et le survivant est v_3 car $v_3\zeta^{AC} = v_0$.
3. enfin, on définit $X = (X_0 \setminus \{v_0\}) \cup (X_1 \setminus \{v_3\})$. Les simplexes de $([v_0]^{AC} \cup [v_1]^{AC}) \setminus X = \{v_0, v_3\}$ sont identifiés entre eux dans (I^{BD}, ζ^{BD}). Le simplexe survivant est v_0.

Toutes les classes d'équivalence de (I^{AC}, ζ^{AC}) différentes de $[v_0]^{AC}$ et $[v_1]^{AC}$ se retrouvent en l'état dans (I^{BD}, ζ^{BD}), et $\sigma\zeta^{BD} = \sigma\zeta^{AC}$ pour les simplexes σ de ces classes.

3.2.1-c Exemple

Ce paragraphe est un exemple de répartition d'identification appliquée à un ensemble semi-simplicial, couvrant l'ensemble des branches de l'algorithme. L'exemple considéré est illustré sur la figure 3.8.

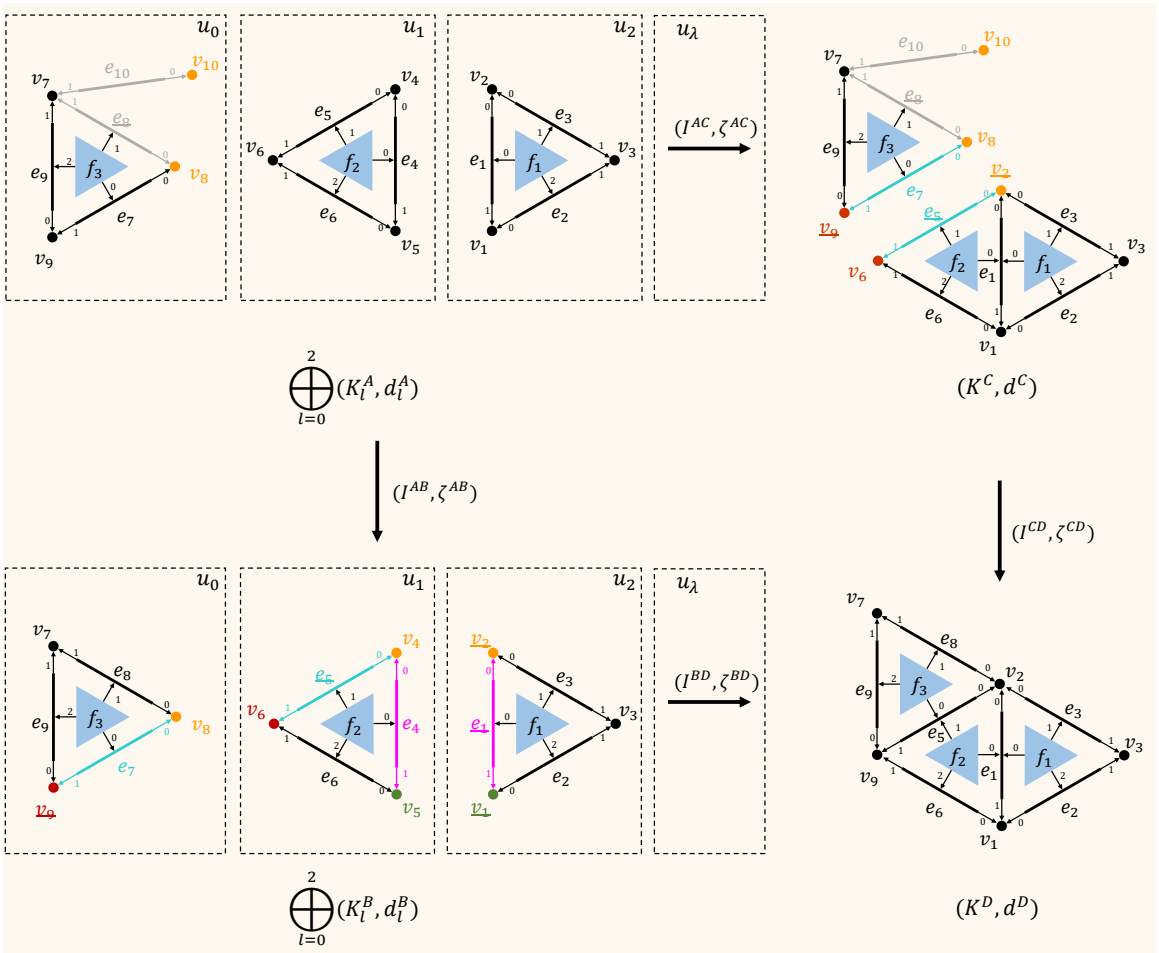


FIGURE 3.8 – La répartition d'une identification (I^{CD}, ζ^{CD}), dans laquelle plusieurs simplexes sont identifiés entre eux. Les simplexes identifiés dans (I^{CD}, ζ^{CD}), (I^{AB}, ζ^{AB}) et (I^{BD}, ζ^{BD}) sont mis en évidence en couleur sur (K^C, d^C), $\oplus_{l=0}^{2}(K_l^A, d_l^A)$ et $\oplus_{l=0}^{2}(K_l^B, d_l^B)$. Pour une identification donnée, les simplexes d'une même classe d'équivalence sont de même couleur. Les simplexes survivants sont surlignés. L'identification (I^{AC}, ζ^{AC}) n'est pas mise en évidence par souci de clarté. Elle consiste à identifier v_1 avec v_5, v_2 avec v_4 et e_1 avec e_4, et les simplexes v_2, v_1 et e_1 sont survivants.

Données de départ. La répartition $\oplus_{l=0}^{2}(K_l^A, d_l^A)$ et les identifications (I^{AC}, ζ^{AC}) et (I^{CD}, ζ^{CD}) sont données. Elles sont caractérisées par :

$$I^{AC} = \{\{v_1, v_5\}, \{v_2, v_4\}, \{e_1, e_4\}\} \quad I^{CD} = \{\{v_2, v_8, v_{10}\}, \{v_9, v_6\}, \{e_5, e_7\}, \{e_8, e_{10}\}\}$$

Les simplexes survivants des classes de (I^{AC}, ζ^{AC}) sont v_1, v_2 et e_1. Les simplexes survivants des classes de (I^{CD}, ζ^{CD}) sont v_2, v_9, e_5 et e_8. Les applications ζ^{AC} et ζ^{CD} en découlent.

Calculs. Pour répartir (I^{CD}, ζ^{CD}), nous parcourons les classes d'équivalence de I^{CD} et on leur applique les 3 étapes de calcul de l'algorithme :

— considérons $[v_2]^{CD} = \{v_2, v_8, v_{10}\}$,
  1. $[v_2]^{AC} \cup [v_8]^{AC} \cup [v_{10}]^{AC} = \{v_2, v_4, v_8, v_{10}\}$;
  2. la restriction de $[v_2]^{AC} \cup [v_8]^{AC} \cup [v_{10}]^{AC}$ par unité de calcul donne :
     · $X_0 = \{v_8, v_{10}\}$. Le survivant est v_8, choisi arbitrairement car $v_8\zeta^{AC} = v_8$, $v_{10}\zeta^{AC} = v_{10}$ et $v_8 \neq v_2$, $v_{10} \neq v_2$;
     · $X_1 = \{v_4\}$;
     · $X_2 = \{v_2\}$;
  3. $X = (X_0 \setminus \{v_8\}) \cup (X_1 \setminus \{v_4\}) \cup (X_2 \setminus \{v_2\}) = \{v_{10}\}$. Les simplexes de $([v_2]^{AC} \cup [v_8]^{AC} \cup [v_{10}]^{AC}) \setminus X = \{v_2, v_4, v_8\}$ sont identifiés entre eux dans (I^{BD}, ζ^{BD}), et v_2 est survivant.

— considérons $[v_9]^{CD} = \{v_9, v_6\}$,
  1. $[v_9]^{AC} \cup [v_6]^{AC} = \{v_9, v_6\}$;
  2. la restriction de $[v_9]^{AC} \cup [v_6]^{AC}$ par unité de calcul donne :
     · $X_0 = \{v_9\}$;
     · $X_1 = \{v_6\}$;
     · $X_2 = \emptyset$.
  3. $X = (X_0 \setminus \{v_9\}) \cup (X_1 \setminus \{v_6\}) \cup X_2 = \emptyset$. Les simplexes de $([v_9]^{AC} \cup [v_6]^{AC}) \setminus X = \{v_9, v_6\}$ sont identifiés entre eux dans (I^{BD}, ζ^{BD}) et v_9 est survivant.

— considérons $[e_5]^{CD} = \{e_5, e_7\}$,
  1. $[e_5]^{AC} \cup [e_7]^{AC} = \{e_5, e_7\}$;
  2. la restriction de $[e_5]^{AC} \cup [e_7]^{AC}$ par unité de calcul donne :
     · $X_0 = \{e_7\}$;
     · $X_1 = \{e_5\}$;
     · $X_2 = \emptyset$.
  3. $X = \emptyset$. Les simplexes de $([e_5]^{AC} \cup [e_7]^{AC}) \setminus X = \{e_5, e_7\}$ sont identifiés entre eux dans (I^{BD}, ζ^{BD}) et e_5 est survivant.

— considérons $[e_8]^{CD} = \{e_8, e_{10}\}$,
  1. $[e_8]^{AC} \cup [e_{10}]^{AC} = \{e_8, e_{10}\}$;
  2. la restriction de $[e_8]^{AC} \cup [e_{10}]^{AC}$ par unité de calcul donne :
     · $X_0 = \{e_8, e_{10}\}$. Le simplexe survivant est e_8 car $e_8\zeta^{AC} = e_8$;
     · $X_1 = \emptyset$;
     · $X_2 = \emptyset$.
  3. $X = (X_0 \setminus \{e_8\}) \cup X_1 \cup X_2 = \{e_{10}\}$. Il reste $([e_8]^{AC} \cup [e_{10}]^{AC}) \setminus X = \{e_8\}$, la répartition de $[e_8]^{CD}$ n'induit donc pas d'identifications dans (I^{BD}, ζ^{BD}).

Enfin, les classes de I^{AC} non impliquées dans une répartition se retrouvent en l'état dans I^{BD}. En particulier :

— $[e_1]^{BD} = [e_1]^{AC} = \{e_1, e_4\}$ et $e_1\zeta^{BD} = e_4\zeta^{BD} = e_1\zeta^{AC} = e_4\zeta^{BD} = e_1$;
— $[v_1]^{BD} = [v_1]^{AC} = \{v_1, v_5\}$ et $v_1\zeta^{BD} = v_5\zeta^{BD} = v_1\zeta^{AC} = v_5\zeta^{BD} = v_1$.

3.2.2 Implications sur les suites exactes courtes effectives

Notons S^{XY} la SECE induite par une identification (I^{XY}, ζ^{XY}). Nous avons vu que, pour calculer l'homologie de l'objet distribué, nous utilisons le théorème SECE (sur S^{AC} pour l'étape t, sur S^{BD} pour l'étape $t+1$). Les 4 identifications impliquées dans une répartition sont représentées sous la forme de SECE. Nous étudions les implications de la répartition d'une identification sur ces dernières. En particulier, nous montrons que S^{AB} et S^{BD} se calculent par modification de S^{AC} et S^{CD}. La figure 3.9 illustre les 4 SECE impliquées dans la répartition de (I^{CD}, ζ^{CD}). Nous commençons par rappeler en quoi une identification (I^{XY}, ζ^{XY}) induit S^{XY}, et comment S^{XY} s'interprète par rapport à (I^{XY}, ζ^{XY}). Ensuite, nous présentons un algorithme de calcul de S^{AB} et S^{BD} à partir de S^{AC} et S^{CD}.

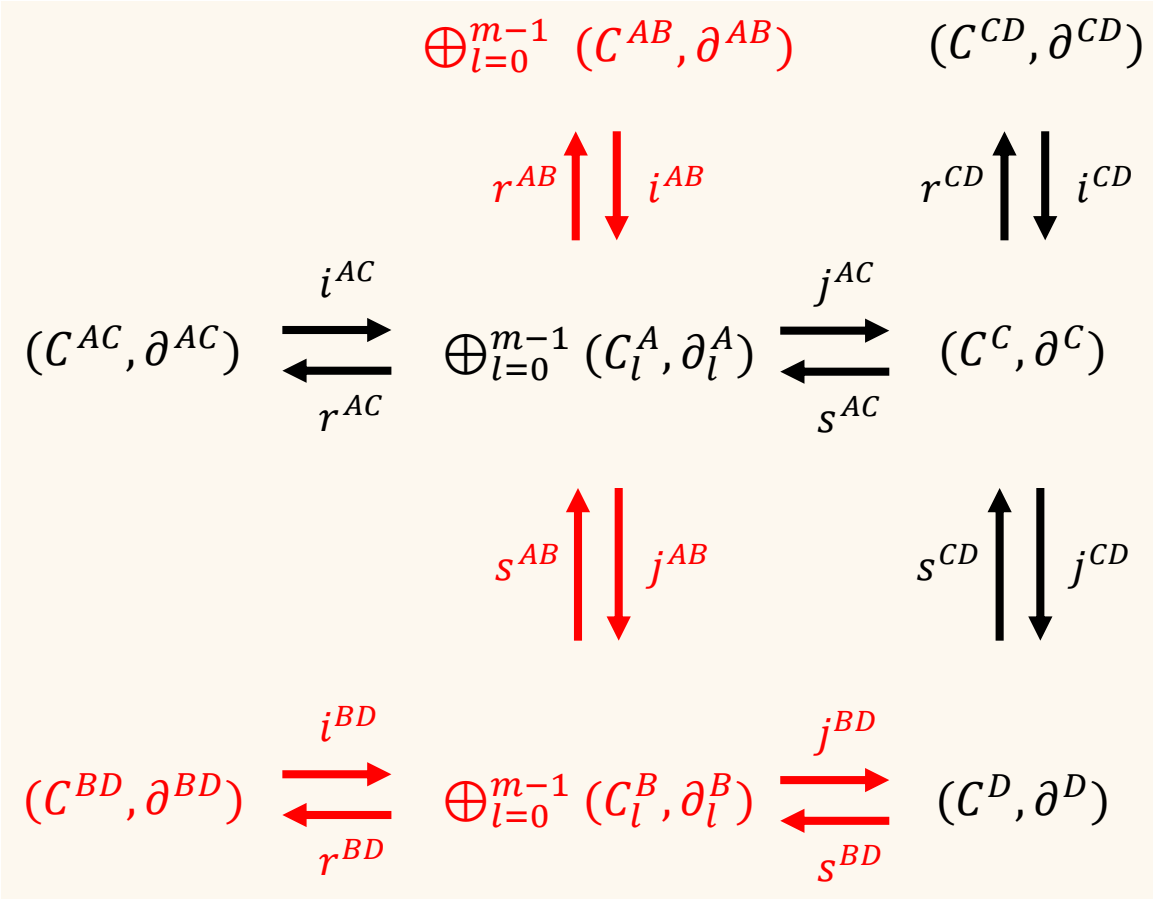


FIGURE 3.9 – Vue d'ensemble des éléments des SECE induites par les 4 identifications impliquées dans la répartition de (I^{CD}, ζ^{CD}). En rouge, les SECE calculées à partir de S^{CD} et S^{BD}.

3.2.2-a Rapppel : une identification induit une SECE

Nous avons vu en sous-section 1.5.1 que toute identification induit une SECE. À l'inverse, les éléments d'une telle SECE peuvent être interprétés pour déduire l'identification dont elle émane. En particulier, le morphisme i encode à lui seul la totalité des informations qui caractérise l'identification. Soit S^{XY} une SECE induite par une identification (I^{XY}, ζ^{XY}), telle que :

$$S^{XY} : (C^{XY}, \partial^{XY}) \underset{r^{XY}}{\overset{i^{XY}}{\rightleftarrows}} (C^X, \partial^X) \underset{s^{XY}}{\overset{j^{XY}}{\rightleftarrows}} (C^Y, \partial^Y)$$

L'interprétation des éléments de S^{XY} par rapport à (I^{XY}, ζ^{XY}) est :

— le complexe de chaînes (C^X, ∂^X) est induit par l'ensemble semi-simplicial avant identification ;

— le complexe de chaînes (C^Y, ∂^Y) est induit par l'ensemble semi-simplicial après identification ;

— le complexe de chaînes (C^{XY}, ∂^{XY}) est engendré par l'ensemble des simplexes non-survivants de (I^{XY}, ζ^{XY}). Dit autrement, tout générateur σ de (C^{XY}, ∂^{XY}) "représente" un simplexe identifié avec au moins un autre simplexe dans (I^{XY}, ζ^{XY}) ;

— l'image de tout générateur σ de (C^{XY}, ∂^{XY}) par i^{XY} est telle que $\sigma i^{XY} = \sigma' - \sigma$, où,

  · σ et σ' appartiennent à la même classe d'équivalence $[\sigma']^{XY}$ de (I^{XY}, ζ^{XY}) ;

  · $\sigma' \zeta^{XY} = \sigma \zeta^{XY} = \sigma'$. Dit autrement, σ' est le simplexe survivant de la classe $[\sigma']^{XY}$.

La SECE (S^{XY}, ζ^{XY}) découle donc entièrement de i^{XY}. Nous utilisons cette observation pour simplifier la sous-section 3.2.2-b, en la limitant au calcul de i^{AB} et i^{BD} par modification de i^{CD} et i^{AC}.

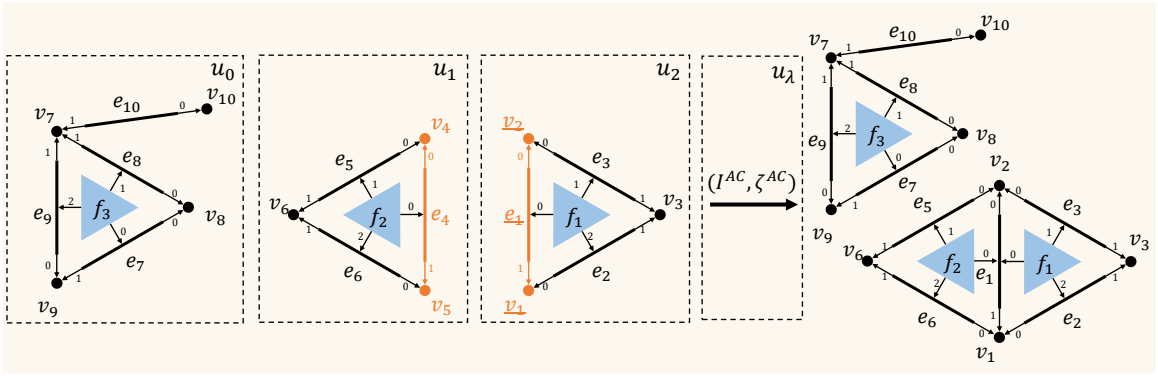
Exemple. La figure 3.10 illustre l'identification (I^{AC}, ζ^{AC}) de la répartition illustrée sur la figure 3.8 et la SECE S^{AC} qu'elle induit. Les générateurs de C^{AC} sont v_4, v_5 et e_4. Le morphisme $i^{AC} : B_{C^{AC}} \to \oplus_{l=0}^{m-1} B_{C_l^A}$ est défini par :

— $v_4 i^{AC} = v_2 - v_4$;

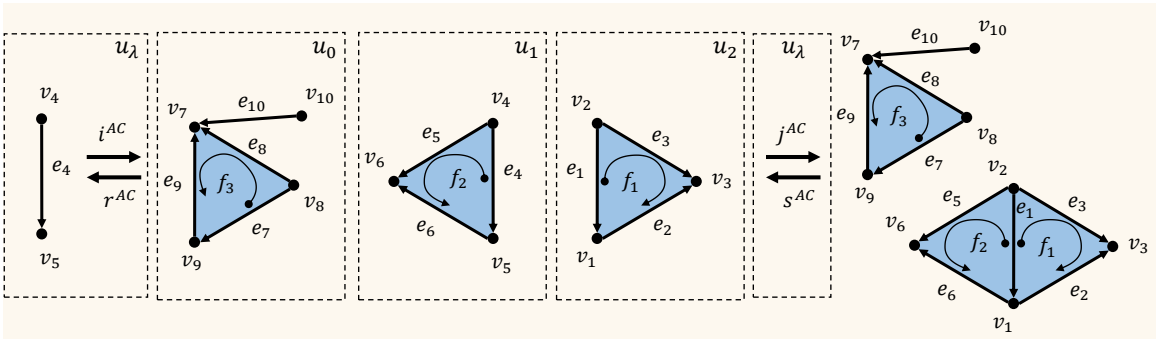
— $v_5 i^{AC} = v_1 - v_5$;

— $e_4 i^{AC} = e_1 - e_4$.

On en déduit les classes d'équivalence des simplexes identifiés dans (I^{AC}, ζ^{AC}) : $[v_2]^{AC} = \{v_2, v_4\}$, $[v_1]^{AC} = \{v_1, v_5\}$ et $[e_4]^{AC} = \{e_1, e_4\}$. En particulier, les simplexes v_2, v_1 et e_1 sont les simplexes survivants de chacune de ces classes.



(a)



(b)

FIGURE 3.10 – L'identification (I^{AC}, ζ^{AC}) de la figure 3.8 et la SECE qu'elle induit. En orange, les simplexes identifiés. Les simplexes survivants sont soulignés. (a) L'identification (I^{AC}, ζ^{AC}). (b) La SECE induite par (I^{AC}, ζ^{AC}).

3.2.2-b Répartition sur les SECE

Une SECE induite par une identification est entièrement caractérisée par le morphisme i qui la compose. Dans ce paragraphe, nous présentons l'algorithme de calcul des morphismes i^{AB} et i^{BD}, qui composent les SECE S^{AB} et S^{BD}, à partir des morphismes i^{CD} et i^{AC}, qui composent les SECE S^{CD} et S^{AC}. Nous précisons les données que prend cet algorithme en entrée et les calculs qu'il effectue. Au fil de ce paragraphe, nous nous appuyons sur l'exemple de la figure 3.11 qui découle de l'exemple déroulé dans le paragraphe 3.2.1-c.

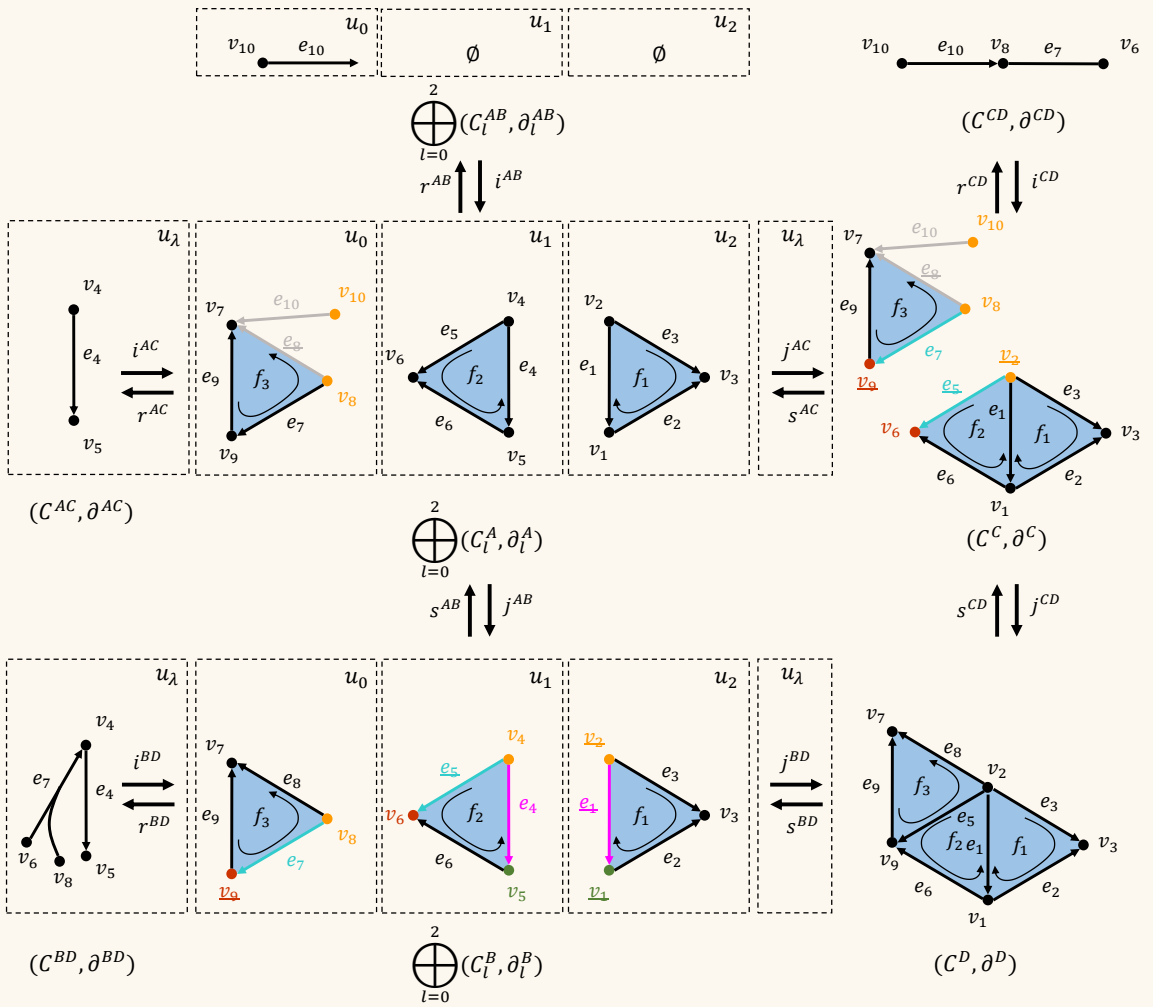


FIGURE 3.11 – Les SECE induites par la répartition de l'identification (I^{CD}, ζ^{CD}) illustrée sur la figure 3.8. Les générateurs de $\oplus_{l=0}^{m-1}(C_l^A, \partial_l^A)$, (C^C, ∂^C) et $\oplus_{l=0}^{m-1}(C_l^B, \partial_l^B)$ sont représentés de la même manière que les simplexes auxquels ils sont associés. Les identifications illustrées sur ces complexes de chaînes sont respectivement (I^{AB}, ζ^{AB}), (I^{CD}, ζ^{CD}) et (I^{BD}, ζ^{BD}).

Données de départ. La SECE S^{AC} est une donnée d'entrée. Elle induite par l'identification de reconstruction (I^{AC}, ζ^{AC}) et la distribution de (K^C, d^C) en $\oplus_{l=0}^{m-1}(K_l^A, d_l^A)$. En particulier, nous nommons $\oplus_{l=0}^{m-1}(C_l^A, \partial_l^A)$ les complexes de chaînes induits par la distribution, avec (C_l^A, ∂_l^A) appartenant à u_l. La seconde donnée d'entrée est la SECE S^{CD}. En particulier, rappelons que (K^C, d^C) et (K^D, d^D) sont supposés trop complexes pour être représentés sur une unité de

calcul. Il en découle que les complexes de chaînes (C^C, ∂^C) et (C^D, ∂^D) sont trop complexes pour être représentés sur une unité de calcul. Seul le morphisme i^{CD} de S^{CD} est explicitement représenté, et il sert de point de départ à la répartition.

Exemple. Sur la figure 3.11, la SECE S^{AC} est une donnée. Elle est représentée sur 4 unités de calcul : u_λ, u_0, u_1 et u_2. Le complexe de chaînes (C^C, ∂^C) n'est pas explicitement représenté. La SECE S^{CD} est la seconde donnée d'entrée. Seul i^{CD} est représenté, sur une unité de calcul arbitraire, et sert de point de départ à la répartition. Les morphismes $i^{AC} : B_{C^{AC}} \to \oplus_{l=0}^{m-1} B_{C_l^A}$ et $i^{CD} : B_{C^{CD}} \to B_{C^C}$ sont définis par :

$i^{AC} : B_{C^{AC}} \to \oplus_{l=0}^{m-1} B_{C_l^A}$
$v_4 i^{AC} = v_2 - v_4$
$v_5 i^{AC} = v_1 - v_5$
$e_4 i^{AC} = e_1 - e_4$

$i^{CD} : B_{C^{CD}} \to B_{C^C}$
$v_6 i^{CD} = v_9 - v_6$
$v_8 i^{CD} = v_2 - v_8$
$v_{10} i^{CD} = v_2 - v_{10}$
$e_7 i^{CD} = e_5 - e_7$
$e_{10} i^{CD} = e_8 - e_{10}$

Pour rappel, si (C, ∂) est un complexe de chaînes induit par une structure topologique, la notation B_C désigne la base de (C, ∂) induite par les cellules de la structure. Les morphismes $\partial^{AC} : B_{C^{AC}} \to B_{C^{AC}}$ et $\partial^{CD} : B_{C^{CD}} \to B_{C^{CD}}$ sont définis par :

$\partial^{AC} : B_{C^{AC}} \to B_{C^{AC}}$
$e_4 \partial^{AC} = v_5 - v_4$

$\partial^{CD} : B_{C^{CD}} \to B_{C^{CD}}$
$e_7 \partial^{CD} = -v_6 - v_8$
$e_{10} \partial^{CD} = v_8 - v_{10}$

Calculs. L'objectif est de calculer les morphismes i^{AB} et i^{BD} qui, comme nous l'avons vu dans le paragraphe 3.2.2-a, caractérisent entièrement les SECE S^{AB} et S^{BD}. Intuitivement, pour cela, l'algorithme consiste à "répartir" les générateurs de C^{AC} et C^{CD} entre C^{AB} et C^{BD}, et à définir leur image par i^{AB} ou i^{BD}. Dit autrement, la répartition vérifie $C^{AC} \oplus C^{CD} = C^{AB} \oplus C^{BD}$.

Exemple. Sur la figure 3.11, on a $B_{C^{AC}} = \{v_4, v_5, e_4\}$, $B_{C^{CD}} = \{v_6, v_8, v_{10}, e_7, e_{10}\}$ et $B_{C^{AB}} = \{v_{10}, e_{10}\}$, $B_{C^{BD}} = \{v_4, v_5, v_6, v_8, e_4, e_7\}$. On constate que $B_{C^{AC}} \cup B_{C^{CD}} = B_{C^{AB}} \cup B_{C^{BD}}$, et donc, par extension, la répartition vérifie $C^{AC} \oplus C^{CD} = C^{AB} \oplus C^{BD}$.

La répartition repose sur l'état que prennent les simplexes qui induisent les générateurs des complexes de chaînes dans les identifications (I^{AC}, ζ^{AC}) et (I^{CD}, ζ^{CD}). En effet, un simplexe peut :

- survivre à (I^{AC}, ζ^{AC}) et (I^{CD}, ζ^{CD}) ;
- survivre à (I^{AC}, ζ^{AC}) mais pas à (I^{CD}, ζ^{CD}) ;
- ne pas survivre à (I^{AC}, ζ^{AC}), auquel cas il n'est pas concerné par (I^{CD}, ζ^{CD}) car il n'existe pas dans (K^C, d^C).

Pour encoder ces différents états, nous utilisons deux variables x et y associées aux simplexes de (K^C, d^C) et $\oplus_{l=0}^{2}(K_l^A, d_l^A)$. Ces dernières représentent leurs coordonnées dans une *table de répartition* comme celle illustrée sur la figure 3.1.

$\sigma_{0,0}$	$\sigma_{1,0}$	$\cdots$	$\sigma_{q-1,0}$
$\sigma_{0,1}$	$\sigma_{1,1}$	$\cdots$	$\sigma_{q-1,1}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
σ_{0,k_0}	σ_{1,k_1}	$\cdots$	$\sigma_{q-1,k_{q-1}}$

TABLE 3.1 – Table de répartition d'une classe d'équivalence $[\sigma]^{CD} = \{\sigma_{0,0}, \ldots, \sigma_{q-1,0}\}$ de (I^{CD}, ζ^{CD}). La variable k_x désigne le cardinal de l'ensemble $[\sigma_{x,0}]^{AC}$ pour tout x compris entre 0 et $q-1$ inclus.

Soit $[\sigma]^{CD} = \{\sigma_{0,0} \ldots \sigma_{q-1,0}\}$ une classe d'équivalence de I^{CD}. Sa table de répartition est telle que :

— la première ligne représente $[\sigma]^{CD}$;

— la x-ème colonne représente $[\sigma_{x,0}]^{AC}$ pour tout x compris entre 0 et $q-1$ inclus.

Soit σ un simplexe de (K^C, d^C) ou de $\oplus_{l=0}^{2}(K_l^A, d_l^A)$. On le note $\sigma_{x,y}$, où :

— x est l'indice du simplexe $\sigma_{x,0}$, survivant de $[\sigma]^{AC}$, dans $[\sigma_{x,0}]^{CD}$, avec $0 \leq x \leq q-1$. Par convention, si $x = 0$, $\sigma_{x,0}$ est survivant dans (I^{CD}, ζ^{CD}) ;

— y est son indice dans $[\sigma]^{AC}$, avec $0 \leq y \leq k_{q-1}$. Par convention, si $y = 0$, σ est survivant dans (I^{AC}, ζ^{AC}).

Observons qu'avec une telle notation, les simplexes de (K^C, d^C) sont nécessairement de la forme $\sigma_{x,0}$, et ceux de (K^D, d^D) sont de la forme $\sigma_{0,0}$. Par abus de notation, nous donnons les mêmes noms (et donc les mêmes coordonnées) aux générateurs des complexes de chaînes que ceux des simplexes auxquels ils sont associés. En particulier, d'après l'hypothèse (iii), notons que chaque colonne d'une table contient au plus un générateur de (C_l^A, ∂_l^A). Pour les morphismes i^{AC} et i^{CD}, la notation implique que :

— les générateurs de C^{CD} sont induits par les simplexes non-survivants de (I^{CD}, ζ^{CD}), c'est-à-dire de la forme $\sigma_{x,0}$ avec $x \neq 0$. En particulier,

$$\sigma_{x,0} i^{CD} = \sigma_{0,0} - \sigma_{x,0} \qquad \forall x \neq 0$$

— les générateurs de C^{AC} sont induits par les simplexes non-survivants de (I^{AC}, ζ^{AC}), c'est-à-dire de la forme $\sigma_{x,y}$ avec $y \neq 0$. En particulier,

$$\sigma_{x,y} i^{AC} = \sigma_{x,0} - \sigma_{x,y} \qquad \forall y \neq 0$$

Exemple. Considérons la classe d'équivalence $[v_2]^{CD} = \{v_2, v_8, v_{10}\}$ et rappelons que les classes d'équivalence de ses simplexes dans I^{AC} sont $[v_2]^{AC} = \{v_2, v_4\}$, $[v_8]^{AC} = \{v_8\}$ et $[v_{10}]^{AC} = \{v_{10}\}$. Les coordonnées de v_2, v_4, v_8 et v_{10} sont :

— pour v_2, $x = 0$ et $y = 0$, ce qui traduit le fait que v_2 est survivant dans (I^{AC}, ζ^{AC}) et (I^{CD}, ζ^{CD}) ;

— pour v_4, $x = 0$ et $y = 1$, ce qui traduit le fait que v_4 est non-survivant dans (I^{AC}, ζ^{AC}) et donc n'existe pas dans (I^{CD}, ζ^{CD}) ;

— pour v_8, $x = 1$ et $y = 0$, ce qui traduit le fait que v_8 est survivant dans (I^{AC}, ζ^{AC}) mais pas dans (I^{CD}, ζ^{CD}) ;

— pour v_{10}, $x = 2$ et $y = 0$, ce qui traduit le fait que v_{10} est survivant dans (I^{AC}, ζ^{AC}) mais pas dans (I^{CD}, ζ^{CD}).

La table de répartition de $[v_2]^{CD} = \{v_2, v_8, v_{10}\}$ est :

v_2	v_8	v_{10}
v_4		

Les morphismes i^{AB} et i^{BD} sont calculés à partir des morphismes i^{AC} et i^{CD} en minimisant les changements d'état des générateurs : autant que possible, un générateur survivant reste un générateur survivant, et un générateur non-survivant reste un générateur non-survivant. Pour cela, observons que la table de répartition d'une classe $[\sigma]^{CD}$ de I^{CD} se décompose en 4 parties illustrées sur la figure 3.12 :

— le simplexe $\sigma_{0,0}$, survivant dans (I^{AC}, ζ^{AC}) et (I^{CD}, ζ^{CD}), au croisement de la première ligne et de la première colonne. Il apparaît avec un coefficient 1 dans i^{AC} et i^{CD} (cf. paragraphe 3.2.2-a si cela n'est pas évident) ;

— les simplexes survivants dans (I^{AC}, ζ^{AC}) et non-survivants dans (I^{CD}, ζ^{CD}). Ils sont sur la première ligne, à l'exception de $\sigma_{0,0}$. Ils apparaissent avec un coefficient 1 dans i^{AC} et avec un coefficient -1 dans i^{CD} ;

— les simplexes σ non-survivants dans (I^{AC}, ζ^{AC}) identifiés avec $\sigma_{0,0}$. Ils sont sur la première colonne, de la forme $\sigma_{0,y}$. Ils apparaissent dans i^{AC} avec un coefficient -1 ;

— les simplexes non-survivants dans (I^{AC}, ζ^{AC}) identifiés avec un autre simplexe que $\sigma_{0,0}$. Ils sont sur toutes les lignes et colonnes à l'exception des premières, de la forme $\sigma_{x,y}$ avec $x, y \neq 0$. Ils apparaissent avec un coefficient -1 dans i^{AC}.

i^{AB} et i^{BD} se calculent par modification de i^{AC} et i^{CD}. Pour minimiser les modifications, il convient de choisir les survivants de (I^{AB}, ζ^{AB}) en priorité sur la première ligne ou sur la première colonne de la table. En effet, soit σ un tel simplexe autre que $\sigma_{0,0}$, on observe que :

— σ est identifié avec $\sigma_{0,0}$ et induit soit $\sigma i^{AC} = \sigma_{0,0} - \sigma$, soit $\sigma i^{CD} = \sigma_{0,0} - \sigma$;

— le seul simplexe à survivre à toutes les identifications est $\sigma_{0,0}$. Dit autrement, $\sigma_{0,0}$ apparaît toujours avec un coefficient 1 ;

— faire survivre σ dans (I^{AB}, ζ^{AB}) induit $\sigma i^{BD} = \sigma i^{AC}$ ou $\sigma i^{BD} = \sigma i^{CD}$, c'est-à-dire que l'image par i^{BD} de σ est définie sans modifications.

.

$\sigma_{0,0}$	$\sigma_{1,0}$	⋯	$\sigma_{q-1,0}$
$\sigma_{0,1}$	$\sigma_{1,1}$	⋯	$\sigma_{q-1,1}$
⋮	⋮	⋱	⋮
σ_{0,k_0}	σ_{1,k_1}	⋯	$\sigma_{q-1,k_{q-1}}$

FIGURE 3.12 – Les 4 parties de la table de répartition d'une classe d'équivalence $[\sigma]^{CD} = \{\sigma_{0,0}, \ldots, \sigma_{q-1,0}\}$. En bleu, le seul générateur de la table qui survit à toutes les identifications : $\sigma_{0,0}$. En vert, les générateurs non-survivants dans (I^{CD}, ζ^{CD}) identifiés avec $\sigma_{0,0}$. En rouge, les générateurs non-survivants dans (I^{AC}, ζ^{AC}) identifiés avec $\sigma_{0,0}$. En blanc, les générateurs non-survivants dans (I^{AC}, ζ^{AC}) qui ne sont jamais identifiés directement avec $\sigma_{0,0}$.

Pour chaque classe d'équivalence $[\sigma]^{CD}$ de I^{CD} composée d'au moins deux simplexes, le calcul de i^{AB} et i^{BD} se fait en 3 temps :

1. on calcule la table de répartition de $[\sigma]^{CD}$;
2. on calcule i^{AB}. Pour cela, on restreint la table de répartition de $[\sigma]^{CD}$ aux différentes unités de calcul u_l. Pour chaque valeur de l comprise entre 0 et $m-1$, il reste au plus un élément par colonne. D'abord, on détermine un générateur survivant σ_l^{AB} pour l, en choisissant, par ordre de possibilité :
   — $\sigma_l^{AB} = \sigma_{0,0}$;
   — $\sigma_l^{AB} = \sigma_{x,0}$. Plusieurs générateurs peuvent correspondre, chacun d'eux peut être choisi arbitrairement ;
   — $\sigma_l^{AB} = \sigma_{0,y}$. Du fait de l'hypothèse (iii), au plus un seul générateur peut correspondre ;
   — $\sigma_l^{AB} = \sigma_{x,y}$. Plusieurs générateurs peuvent correspondre, chacun d'eux peut être choisi arbitrairement.

   Ensuite, pour chaque générateur σ différent de σ_l^{AB} et appartenant à la restriction de la table de répartition à u_l, on distingue :
   — le cas où σ est de la forme $\sigma_{x,0}$. Dans ce cas, il existe $\sigma i^{CD} = \sigma_{0,0} - \sigma$ que l'on modifie en $\sigma i^{AB} = \sigma_l^{AB} - \sigma$;
   — le cas où σ est de la forme $\sigma_{0,y}$. Dans ce cas, il existe $\sigma i^{AC} = \sigma_{0,0} - \sigma$ que l'on modifie en $\sigma i^{AB} = \sigma_l^{AB} - \sigma$;
   — le cas où σ est de la forme $\sigma_{x,y}$. Dans ce cas, il existe $\sigma i^{AC} = \sigma_{x,0} - \sigma$ que l'on modifie en $\sigma i^{AB} = \sigma_l^{AB} - \sigma$.
3. on calcule i^{BD} une fois toutes les valeurs de l traitées. Pour ce faire, on restreint la table de répartition aux générateurs survivants σ_l^{AB} précédemment définis, pour l compris entre 0 et $m-1$. Pour chaque σ_l^{AB} différent de $\sigma_{0,0}$, on distingue :
   — le cas où σ_l^{AB} est de la forme $\sigma_{x,0}$. Dans ce cas, il existe $\sigma_l^{AB} i^{CD} = \sigma_{0,0} - \sigma_l^{AB}$ que l'on modifie en $\sigma i^{BD} = \sigma_{0,0} - \sigma_l^{AB}$;
   — le cas où σ_l^{AB} est de la forme $\sigma_{0,y}$. Dans ce cas, il existe $\sigma_l^{AB} i^{AC} = \sigma_{0,0} - \sigma_l^{AB}$ que l'on modifie en $\sigma i^{BD} = \sigma_{0,0} - \sigma_l^{AB}$;
   — le cas où σ_l^{AB} est de la forme $\sigma_{x,y}$. Dans ce cas, il existe $\sigma_l^{AB} i^{AC} = \sigma_{x,0} - \sigma_l^{AB}$ que l'on modifie en $\sigma i^{BD} = \sigma_{0,0} - \sigma_l^{AB}$.

Les classes d'équivalence de (I^{AC}, ζ^{AC}) non traitées une fois toutes les classes d'équivalence de (I^{CD}, ζ^{CD}) traitées ne sont pas modifiées. Dit autrement, la partie de i^{AC} concernée par ces classes d'équivalence se retrouve en l'état dans i^{BD}. Soit $[\sigma]^{AC}$ une telle classe, où σ est le simplexe survivant :

— il existe $\sigma' \in [\sigma]^{AC}$ tel que $\sigma' i^{AC} = \sigma - \sigma'$, avec $\sigma' \neq \sigma$;
— $\sigma' i^{AC} = \sigma - \sigma'$ se retrouve en l'état dans i^{BD}, c'est-à-dire que $\sigma' i^{BD} = \sigma - \sigma'$.

Exemple. Considérons la classe d'équivalence $[v_2]^{CD}$ de I^{CD}. Sa répartition se fait en trois temps :

1. on calcule la table de répartition de $[v_2]^{CD}$. Celle-ci est présentée dans l'exemple qui précède ;
2. on calcule i^{AB} en restreignant cette table aux différentes u_l pour l compris entre 0 et 2. On a, dans l'ordre :

	v_8	v_{10}

v_4		

v_2		

   Pour chaque unité de calcul, on détermine un générateur survivant :

— $\sigma_{l=0}^{AB} = v_8$, que l'on choisit arbitrairement entre v_{10} et v_8 car ils sont tout deux de la forme $\sigma_{x,0}$;

— $\sigma_{l=1}^{AB} = v_4$;

— $\sigma_{l=2}^{AB} = v_2$.

Ensuite, on constate que seule la restriction de la table à u_0 est composée de plus d'un générateur. En particulier, v_{10} est le seul générateur de u_0 différent de $\sigma_{l=0}^{AB} = v_8$. On a :

— $v_{10}i^{CD} = v_2 - v_{10}$ que l'on modifie en $v_{10}i^{AB} = v_8 - v_{10}$ car v_{10} est de la forme $\sigma_{x,0}$.

3. on calcule i^{BD} une fois toutes les valeurs de l traitées. Pour ce faire, on restreint la table de répartition de $[v_2]^{CD}$ aux générateurs survivants σ_l^{AB} précédemment définis, pour l compris entre 0 et 2. On obtient :

v_2	v_8	
v_4		

Les générateurs σ_l^{AB} différent de $\sigma_{0,0} = v_2$ sont :

— v_4, de la forme $\sigma 0, y$. Il existe donc $v_4 i^{AC} = v_2 - v_4$ que l'on modifie en $v_4 i^{BD} = v_2 - v_4$;

— v_8, de la forme $\sigma_{x,0}$. Il existe donc $v_8 i^{CD} = v_2 - v_8$ que l'on modifie en $v_8 i^{BD} = v_2 - v_8$.

En appliquant ce même algorithme aux classes d'équivalence $[v_9]^{CD}$, $[e_5]^{CD}$ et $[e_8]^{CD}$ de I^{CD}, on arrive à :

$i^{AB} : \oplus_{l=0}^{m-1} B_{C_l^A} \to \oplus_{l=0}^{m-1} B_{C_l^B}$
$v_{10}i^{AB} = v_8 - v_{10}$
$e_{10}i^{AB} = e_8 - e_{10}$

$i^{BD} : B_{C^{BD}} \to \oplus_{l=0}^{m-1} B_{C_l^B}$
$v_4 i^{BD} = v_2 - v_4$
$v_8 i^{BD} = v_2 - v_8$
$v_{10}i^{BD} = v_2 - v_{10}$
$v_6 i^{BD} = v_9 - v_6$
$e_4 i^{BD} = e_1 - e_4$
$e_7 i^{BD} = e_5 - e_7$

En particulier, $[v_1]^{AC}$ et $[e_1]^{AC}$ sont des classes d'équivalence de (I^{AC}, ζ^{AC}) non traitée par l'algorithme, d'où le fait que :

— $v_4 i^{AC} = v_2 - v_4$ soit modifié en $v_4 i^{BD} = v_2 - v_4$;

— $e_4 i^{AC} = e_1 - e_4$ soit modifié en $e_4 i^{BD} = e_1 - e_4$.

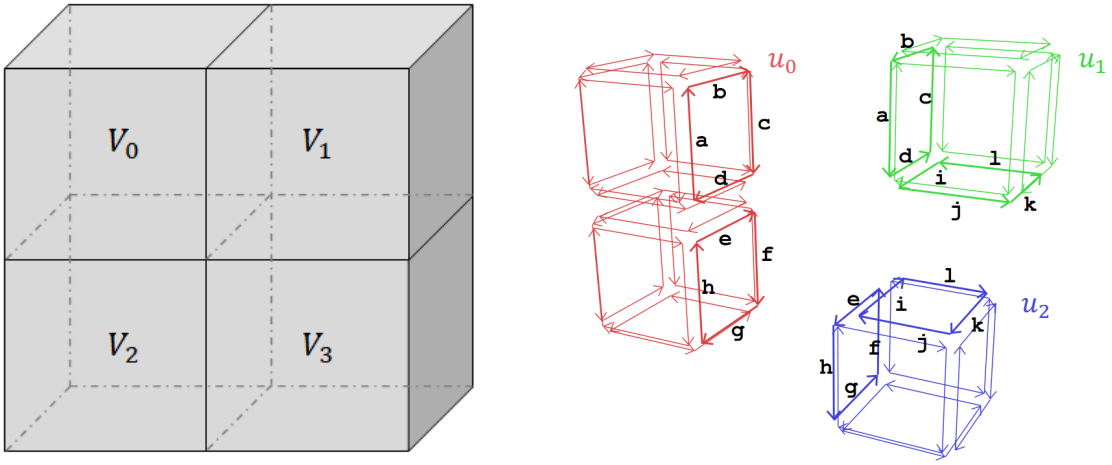
3.3 Implémentation

Pour accélérer le développement, nous sommes partis du code développé dans les travaux de [88] et du logiciel que nous avons développé dans le cadre du Chapitre 2. En particulier, nous réutilisons les répertoires de cellules et la représentation matricielle présentées en sous-section 2.3. À ce jour, le développement du logiciel reste à terminer. En particulier, l'identification de passage d'une étape t à une étape $t+1$ est définie au niveau de la distribution de la carte combinatoire et non de la carte combinatoire distribuée. En conséquence, une partie de l'algorithme de répartition d'identification est triviale : on connaît directement les identifications applicables à la distribution, et celles qui alimentent l'identification de reconstruction. Nous présentons les objectifs de l'implémentation de l'algorithme de répartition d'une identification, les outils que nous utilisons pour le développement et enfin l'architecture logicielle qui en découle.

Remarque. Le code source du projet est disponible à l'adresse `https://gitlab.xlim.fr/hominc/homincdistributed`.

3.3.1 Contexte et objectifs

Contexte. Le point de départ du développement est une base de code issue des travaux de [88]. Les objectifs de ces travaux sont différents de ceux présentés dans le préambule du chapitre. En effet, dans [88], l'objectif est de permettre l'application d'opérations sur des maillages supposés trop complexes pour être manipulés par une unité de calcul, et dont la topologie est représentée par une carte orientée. Il n'est pas question de calcul d'homologie. Une contribution importante



(a) (b)

FIGURE 3.13 – La carte orientée distribuée d'un maillage 3D composé de 4 volumes V_0, V_1, V_3, V_4. Cette figure est issue des Figures 1-a et 5 de [88]. (a) Le maillage. (b) La carte orientée fragmentée en 3 blocs distribués sur 3 unités de calcul u_0, u_1 et u_2. Un label est attribué aux brins critiques, représenté ici par une lettre minuscule.

de [88] est la définition de la notion de carte orientée distribuée. Intuitivement, une carte orientée distribuée fragmente une carte orientée en plusieurs cartes orientées indépendantes plus petites, appelées "blocs", et associe les différents blocs à l'aide d'un système de labels attribués aux *brins*

critiques. Un brin critique est un brin au croisement de deux blocs. La figure 3.13 illustre une carte orientée distribuée. Pour illustrer l'intérêt de cette nouvelle notion, les auteurs étudient les performances d'une subdivision hexaédrique adaptative. Intuitivement, elle consiste à subdiviser une carte orientée en hexaèdres de manière plus ou moins fine par endroit, selon la géométrie du maillage. La figure 3.14 illustre une telle subdivision. Cette subdivision s'applique à une carte orientée distribuée ou non distribuée. Nous avons utilisé cette subdivision pour générer les

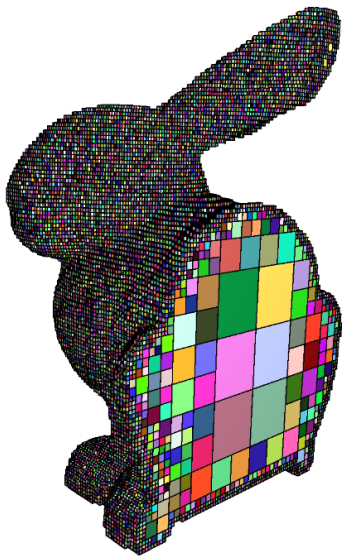


FIGURE 3.14 – Une subdivision hexaédrique adaptative d'un bloc d'une carte orientée distribuée. Cette figure est issue de la Figure 8 de [88].

processus de construction avec lesquels nous travaillons. L'idée est la suivante : étant donnée une carte orientée distribuée subdivisée, on "découd" les hexaèdres des différents blocs que l'on va ensuite recoudre, en définissant par la même occasion une suite d'identifications caractérisant un processus de construction. L'étape initiale de ce processus est l'étape qui précède la première "couture".

Objectifs. Dans [88], l'objectif premier est de permettre la manipulation d'une carte orientée trop complexe pour être manipulée par une unité de calcul. En particulier, la subdivision hexaèdrique adaptative sert d'exemple pour introduire la notion de carte orientée distribuée. Dans notre cadre, le code issue de [88] est utilisé pour calculer une distribution d'une carte orientée, et l'objectif est le calcul de son homologie à mesure que des opérations de couture lui sont appliquées. Pour cela, comme présenté dans le préambule de ce chapitre, on maintient la distribution et une identification de reconstruction sur laquelle on applique le théorème SECE. Il en résulte une équivalence homologique trop complexe pour être représentée sur une unité de calcul ; seul son petit complexe de chaînes est représenté et utilisé pour calculer l'homologie de la carte orientée distribuée. Cela induit notamment l'hypothèse (ii).

3.3.2 Outils

Le code que nous utilisons repose sur l'implémentation des cartes orientées proposée dans la bibliothèque logicielle *Computational Geometry Algorithms Library* (CGAL), qui gère la synchronisation des unités de calcul à l'aide du standard *Message Passing Interface* (MPI). Nous

présentons ces deux outils, que nous avons choisi d'utiliser à notre tour pour l'implémentation de l'algorithme de répartition d'une identification.

3.3.2-a CGAL - paquet LCC

CGAL est une bibliothèque C++ de géométrie qui regroupe un grand nombre de structures de données et d'algorithmes. Elle est généraliste en cela qu'elle ne répond pas à un besoin en particulier mais propose plutôt un ensemble d'outils dont l'utilisateur peut se servir pour répondre à son besoin : on y trouve par exemple des algorithmes de calcul de diagrammes de Voronoi, des algorithmes de reconstruction de surfaces à partir de nuages de points etc ... Ainsi, elle se décompose en plusieurs *paquets* regroupés dans de grandes catégories (arithmétique, combinatoire, reconstruction etc...).

Nous utilisons le paquet *Linear Cell Complex* (LCC) de la catégorie *Cell Complexes and Polyhedra*. Il propose notamment l'implémentation de la notion de carte orientée et des opérations de modifications de ces dernières, dont la "couture" et son inverse.

3.3.2-b MPI

L'une des problématiques liées au développement d'un algorithme parallèle est la synchronisation des unités de calcul travaillant de pair. Plusieurs réponses ont été apportées à cette problématique en distinguant :

— les cas où la mémoire est partagée entre les unités de calcul ;

— les cas où la mémoire est distribuée entre les unités de calcul ;

— les cas où la mémoire est en partie partagée et en partie distribuée entre les unités de calcul.

Le code que nous utilisons relève du second cas, où la mémoire est distribuée entre les unités de calcul. Dans ce cas, une solution pour synchroniser les unités de calcul est de les faire *communiquer* au travers d'*échange de messages*, en respectant un *protocole*. Un standard a été établi pour définir les caractéristiques de ce protocole : *Message Passing Interface* (MPI), maintenu par le groupe MPI Forum [89]. Ce standard définit un ensemble de fonctionnalités que doit garantir toute implémentation qui en relève. Parmi les fonctions élémentaires de ce standard, on retrouve :

— le fait que les unités de calcul (appelées *processus*) communiquent par échange de *messages* au travers d'un *réseau* ;

— le fait d'attribuer un *groupe* ainsi qu'un *rang* au sein de ce groupe à chaque unité de calcul ;

— plusieurs façons de faire communiquer les unités de calcul entre elles, dont :

  · la communication point à point, où une unité s'adresse à une seule autre unité au travers de son rang ;

  · la communication "collective", où une unité s'adresse à toutes les unités d'un ou plusieurs groupes.

Il existe plusieurs bibliothèques logicielles implémentant ce standard. Dans le cadre de ces travaux, nous nous sommes servis de deux d'entre elles : OpenMPI pour Ubuntu et MSMPI pour Windows.

3.3.3 Architecture

Dans cette sous-section, nous présentons "l'architecture" du logiciel, c'est-à-dire la manière dont sont réparties les données impliquées dans la répartition d'une identification entre les différentes unités de calcul, ainsi que les messages que ces dernières s'échangent.

Remarque. Le développement n'étant pas terminé, l'objectif de la sous-section est simplement de faire état de l'architecture que nous utilisons à ce jour. Il est possible que celle-ci soit modifiée dans le futur.

Données par unité de calcul. La carte orientée est distribuée sur m unités de calcul sous la forme de "blocs", c'est-à-dire de cartes orientées indépendantes les unes des autres. Nous nommons ces unités de calcul u_l pour l compris entre 0 et $m-1$. Les données que contient u_l sont :

— (B_l, α_l), le bloc de u_l ;

— le complexe de chaînes (C_l^A, ∂_l^A) induit par (B_l, α_l) ;

— l'équivalence homologique γ_l^A associée à (C_l^A, ∂_l^A).

En particulier, de par la définition d'une carte orientée distribuée, les complexes de chaînes (C_l^A, ∂_l^A) sont indépendants les uns des autres, c'est-à-dire que le bord de tout générateur de (C_l^A, ∂_l^A) appartient à (C_l^A, ∂_l^A). Ces données sont illustrées sur la figure 3.15.

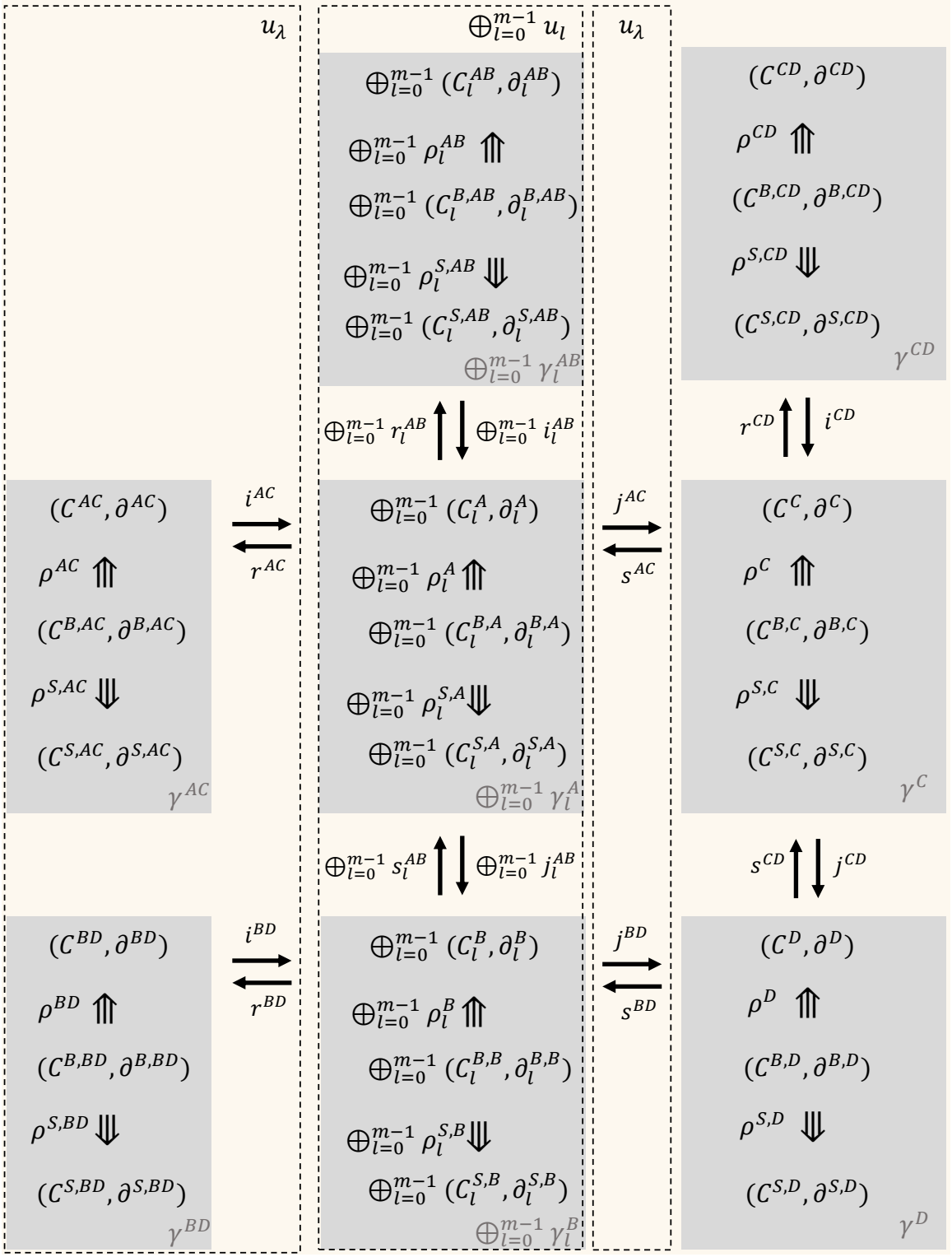


FIGURE 3.15 – Les données des différentes unités de calcul u_l et de l'unité de calcul critique u_λ lors de la répartition d'une identification, pour l compris entre 0 et $m-1$. Les cartes orientées ne sont pas représentées. Les équivalences homologiques sont mises en évidence par un fond gris.

Unité de calcul critique. Comme nous l'avons vu dans la sous-section 3.3.1, on suppose initialement que l'identification de reconstruction est vide et que celle-ci est alimentée au fil des identifications induites par les coutures appliquées à la carte orientée. Ces mises à jour sont effectuées par une unité de calcul qui peut-être :

— une unité de calcul parmi celles contenant les blocs ;

— une unité de calcul u_λ spécifiquement dédiée à cette tâche.

Nous avons choisi d'utiliser une unité de calcul critique u_λ, et utilisons donc au total $m+1$ unités de calcul. u_λ contient l'identification de reconstruction de chaque étape. La représentation de l'identification de reconstruction est particulière car les cellules qu'elle implique appartiennent à des unités de calcul distinctes. Nous avons choisi d'exprimer ces cellules sous la forme de couples `(PID, cellID)`, où `PID` désigne l'identifiant l de l'unité de calcul à laquelle appartient la cellule `cellID`, qui désigne l'indice de cette dernière dans u_l.

Répartition d'une identification. Lorsque la carte orientée distribuée subit une opération de couture, elle induit une identification (I^{CD}, ζ^{CD}) qui est alors répartie. En particulier, nous avons vu dans le paragraphe 3.2.2-b que cela implique de calculer deux SECE S^{AB} et S^{BD}. Plus précisément :

— la SECE S^{AB} peut être décomposée par unité de calcul u_l en plusieurs SECE S_l^{AB}. Pour calculer γ_l^B, chaque u_l applique le théorème SECE sur S_l^{AB} ;

— la SECE S^{BD} n'est jamais explicitement calculée. L'unité de calcul critique u_λ contient simplement l'identification de reconstruction, stockée sous la forme de classes d'équivalences, qu'elle modifie en fonction de ce que doit être S^{BD}. Cette identification n'est utilisée que lorsque l'on souhaite calculer l'homologie de la carte orientée distribuée.

À ce jour, la répartition se limite à ces calculs. En particulier, pour l'instant, aucune unité de calcul n'est chargée de définir les cellules identifiées dans les différentes S_l^{AB} ou celles qui alimentent l'identification de reconstruction. Cela découle du fait que l'opération de couture est définie sur la distribution de la carte orientée, et donc chaque unité de calcul sait directement si elle est concernée par la couture ou non.

Exemple. Sur la figure 3.16, la SECE S^{AB} est décomposée par unité de calcul en S_0^{AB}, S_1^{AB} et S_2^{AB}. Observons notamment que :

— seule γ_0^B est modifiée par rapport à γ_0^A, et que γ_1^B et γ_2^B sont respectivement égales à γ_1^A et γ_2^A ;

— la SECE S^{BD} n'est jamais explicitement calculée. En revanche, u_λ contient l'identification de reconstruction (I^{AC}, ζ^{AC}) qu'elle met à jour en (I^{BD}, ζ^{BD}).

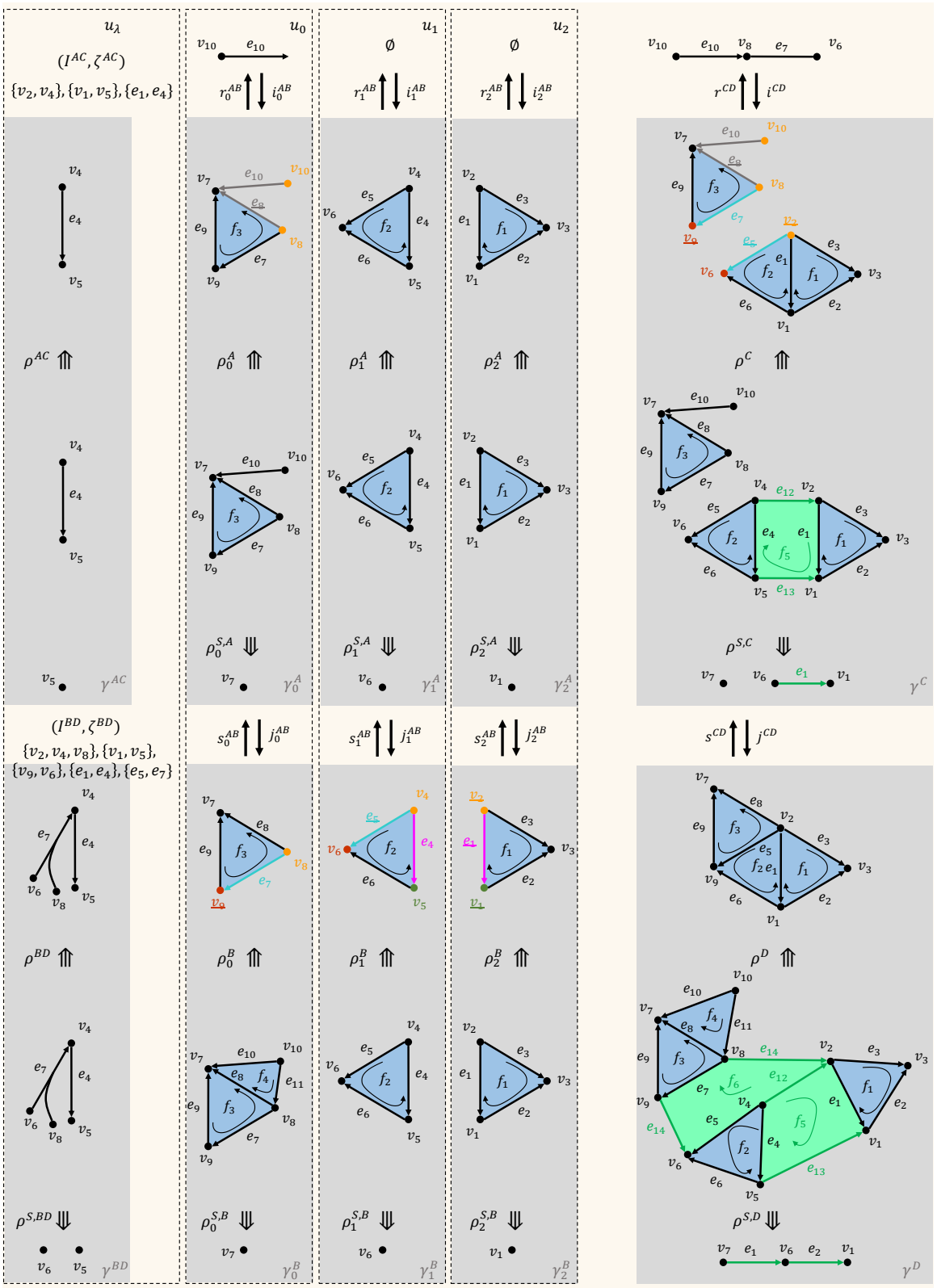


FIGURE 3.16 – Les données des différentes unités de calcul lors de la répartition de l'identification de la figure 3.11. Les cartes orientées et toutes les équivalences homologiques ne sont pas représentées. En vert fluo, les générateurs ajoutés à $(C^{B,D}, \partial^{B,D})$ et $(C^{S,D}, \partial^{S,D})$ par rapport à $\oplus_{l=0}^{2}(C_l^{B,B}, \partial_l^{B,B})$ et $\oplus_{l=0}^{2}(C_l^{S,B}, \partial_l^{S,B})$ si le théorème SECE était appliqué sur la SECE induite par (I^{BD}, ζ^{BD}).

Calcul de l'homologie de la carte orientée "globale". Pour calculer l'homologie de la carte orientée "globale", l'unité de calcul critique u_λ applique le théorème SECE sur l'identification de reconstruction. La problématique est la suivante : une partie des données nécessaires à l'application du théorème SECE est éclatée sur plusieurs unités de calcul, et certaines d'entre elles ne peuvent pas être manipulées par une seule unité de calcul.

Exemple. Sur la figure 3.16, si l'on applique le théorème SECE à l'identification de reconstruction (I^{AC}, ζ^{AC}), on se retrouve à construire le gros complexe de chaînes de l'équivalence homologique γ^C. Or, d'après l'hypothèse (i), ce complexe de chaînes ne peut pas être représenté sur une seule unité de calcul.

Pour répondre à cette problématique, revenons sur ce que représente une équivalence homologique. Une équivalence homologique associe 3 complexes de chaînes entre eux via 2 réductions. Ces complexes de chaînes ont "la même" homologie, c'est-à-dire qu'ils ont les mêmes nombres de Betti et les mêmes coefficients de torsion, et qu'il est possible d'exprimer les générateurs des groupes d'homologie de l'un d'eux dans la base des deux autres. En particulier, l'un de ces complexes de chaînes est plus petit que les deux autres. Il est la seule donnée utilisée pour calculer l'homologie des 3 complexes de chaînes (via une FNS par exemple, cf. sous-section 1.4.1).

Le théorème SECE sert à maintenir une équivalence homologique γ^t associée à une structure topologique lorsqu'une identification lui est appliquée. En particulier, le petit complexe de chaînes de γ^t est maintenu à l'aide de la formule (cf. sous-sections 1.4.3 et 1.5.1) :

$$\partial^{S,t+1} = \begin{pmatrix} -\partial^S & i^S \\ 0 & \partial^{S,t} \end{pmatrix}$$

Maintenir une équivalence homologique fait sens si elle est utilisée comme point de départ d'une autre application du théorème SECE. Or, lorsque u_λ applique le théorème à l'identification de reconstruction, l'équivalence homologique qui en résulte n'a d'autre intérêt que d'être utilisée pour le calcul de l'homologie de la carte orientée "globale". Elle n'est plus utilisée par la suite. Dit autrement, la seule donnée dont u_λ a besoin est en réalité le petit complexe de chaînes de cette équivalence homologique, qui correspond à $\partial^{S,t+1}$ dans la formule.

Exemple. Sur la figure 3.16, l'équivalence homologique γ^C résulte de l'application du théorème SECE sur (I^{AC}, ζ^{AC}) par u_λ. En particulier, parmi toutes les données de γ^C, seul son petit complexe de chaînes est utilisé pour calculer l'homologie de la carte orientée distribuée.

Le calcul de l'homologie de la carte orientée distribuée revient donc à construire ce petit complexe de chaînes. Pour ce faire, nous procédons en 3 étapes :

1. on récupère $\partial^{S,t}$ sur u_λ. En effet, dans le cas distribué, cette donnée est distribuée sur les différentes unités de calcul u_l et correspond aux petits complexes de chaînes des équivalences homologiques γ_l^A. Cette étape implique des communications :
   - u_λ informe les différentes u_l qu'elle souhaite récupérer leur partie de $\partial^{S,t}$;
   - les différentes u_l lui répondent ;
2. on calcule ensuite $-\partial^S$, qui correspond au petit complexe de chaînes de l'équivalence homologique γ induite par l'identification de reconstruction (γ^{AC} ou γ^{BD} dans notre cas). Pour cela, on commence par calculer $\partial = i\partial^t r$, qui correspond au complexe de chaînes "du haut" de γ. Notons que ∂^t est une donnée qui est distribuée sur les différentes unités de calcul u_l ($\oplus_{l=0}^{m-1}\partial_l^A$ ou $\oplus_{l=0}^{m-1}\partial_l^B$ dans notre cas). Pour ce calcul, on récupère donc d'abord sur u_λ la partie de ∂^t qui représente le bord des cellules impliquées dans

l'identification de reconstruction, puis on calcule ∂ et enfin $-\partial^S$. Cette étape implique des communications dans lesquelles :

— u_λ informe les différentes u_l qu'elle souhaite récupérer la partie de ∂^t qui correspond aux cellules impliquées dans l'identification de reconstruction ;

— les différentes u_l lui répondent ;

3. calculer $i^S = g^S f i g^t f^{S,t}$. Notons que g^t et $f^{S,t}$ sont des données distribuées sur les différentes unités de calcul u_l ($\oplus_{l=0}^{m-1}\gamma_l^A$ ou $\oplus_{l=0}^{m-1}\gamma_l^B$ dans notre cas). On commence donc par calculer $g^t f^{S,t}$ sur les différentes u_l pour toutes les cellules impliquées dans l'identification de reconstruction, puis on récupère ce résultat sur u_λ qui se charge de terminer le calcul de i^S. Cette étape implique donc des communications :

— u_λ informe les différentes u_l qu'elles doivent calculer $g^t f^{S,t}$ pour les cellules impliquées dans l'identification de reconstruction ;

— les différentes u_l lui répondent ;

Messages échangés. On constate que pour construire le petit complexe de chaîne à partir duquel on calcule l'homologie de la carte orientée distribuée, des messages sont échangés entre les unités de calcul. Ils sont, dans l'ordre :

— de u_λ aux différentes u_l. Ces messages contiennent les indices des cellules de u_l impliquées dans l'identification de reconstruction ;

— des différentes u_l à u_λ. Ces messages contiennent $\partial_l^{S,A}$ et l'image des cellules du message précédent par ∂_l et $g^t f^{S,t}$.

À ce jour, ces messages sont les seuls utilisés par l'implémentation. Pour autant, l'étape 3 du développement reste à faire (cf. paragraphe 3.1). Celle-ci implique notamment de dédier une unité de calcul à la répartition d'une identification entre :

— ce qui peut être identifié sur les différentes u_l ;

— ce qui alimente l'identification de reconstruction sur u_λ.

Cette étape impliquera forcément de faire communiquer les unités le calcul.

3.4 Conclusion et perspectives

Conclusion. Nous supposons qu'une structure topologique est trop complexe pour être manipulée par une unité et est donc éclatée en plusieurs morceaux M_l distribués sur plusieurs unités de calcul u_l. Elle est représentée par sa distribution, c'est-à-dire les couples (M_l, u_l), et d'une identification de reconstruction (I^{AC}, ζ^{AC}).

Lorsque la structure topologique distribuée subit une identification (I^{CD}, ζ^{CD}), sa distribution et l'identification de reconstruction ne lui correspondent plus. Nous présentons un algorithme de répartition qui a pour but de calculer une distribution et une identification de reconstruction qui correspondent à la structure topologique après que (I^{CD}, ζ^{CD}) lui soit appliquée. La répartition de (I^{CD}, ζ^{CD}) revient à calculer deux identifications :

— (I^{AB}, ζ^{AB}), dans laquelle les cellules identifiées entre elles appartiennent aux mêmes unités de calcul. Il en résulte la distribution de la structure topologique après que (I^{CD}, ζ^{CD}) lui ait été appliquée ;

— (I^{BD}, ζ^{BD}), dans laquelle les cellules identifiées appartiennent à des unités de calcul distinctes. Elle correspond à l'identification de reconstruction de la structure topologique à partir de la distribution qui résulte de (I^{AB}, ζ^{AB}), et après que (I^{CD}, ζ^{CD}) lui ait été appliquée.

Pour calculer l'homologie de la structure topologique distribuée, nous utilisons le théorème SECE sur la SECE induite par l'identification de reconstruction. Nous montrons qu'il est possible de calculer les SECE S^{AB} et S^{BD}, induites respectivement par (I^{AB}, ζ^{AB}) et (I^{BD}, ζ^{BD}), à partir des morphismes i^{CD} et i^{AC} des SECE S^{CD} et S^{AC}, induites respectivement par (I^{CD}, ζ^{CD}) et (I^{AC}, ζ^{AC}). Nous proposons un algorithme réalisant ce calcul basé sur la notion de table de répartition.

Nous présentons un début d'implémentation de l'algorithme de répartition, utilisant le standard MPI pour gérer la synchronisation des unités de calcul. Nous présentons l'architecture logicielle que nous avons choisie pour manipuler la distribution de la structure topologique et l'identification de reconstruction. Cette dernière est maintenue par une unité de calcul critique u_λ qui lui est dédiée. En particulier, u_λ est en charge du calcul de l'homologie de la structure topologique "globale".

Perspectives d'évolutions. Au cours de ces travaux, nous nous sommes intéressés à la répartition d'une identification. Une première évolution consisterait à définir un algorithme de répartition d'une désidentification. Une fois cela fait, il resterait à étudier ces deux algorithmes dans un processus de construction composé des deux opérations, ce qui permettrait de couvrir la totalité des cas.

Enfin, il reste à terminer le développement du logiciel pour pouvoir proposer une étude expérimentale dans laquelle on s'intéresserait non seulement au temps et à l'espace, mais aussi au coût des communications entre unités de calcul.

Chapitre 4

Théorème SECE et homologie persistante

Contenu du chapitre

4.1 Préambule

Le but de ce chapitre est d'établir des liens entre le théorème SECE et l'homologie persistante [1]. Pour cela, nous mettons en parallèle le théorème SECE et :

— l'homologie persistante "de base" (basée sur la notion filtration, cf. définition 1.4.2). Nous montrons que l'application successive du théorème SECE à l'identification induit une filtration, et que ce n'est pas le cas de la désidentification ;

— l'homologie persistante "de tours". Nous montrons que les opérations élémentaires sur lesquelles elle repose (l'effondrement et l'inclusion), définies sur les complexes simpliciaux, s'expriment sur les ensembles simpliciaux sous la forme d'identification, de désidentification et de dégénérescence. De cette façon, nous établissons un cadre commun à la persistance de tours et au théorème SECE.

4.2 Application du théorème SECE et filtration

Contexte. Rappelons qu'une équivalence homologique

$$\gamma : (C, \partial) \overset{\rho}{\Longleftarrow} (C^B, \partial^B) \overset{\rho^S}{\Longrightarrow} (C^S, \partial^S)$$

contient un complexe de chaînes intermédiaire (C^B, ∂^B) plus "gros" [2] que les deux autres, c'est-à-dire qu'il contient plus de générateurs (cf. définition 1.4.5). Soit γ^t l'équivalence homologique γ à l'étape t d'un processus de construction. Comme nous l'avons vu dans les sous-sections 2.1.3 et 2.1.4, le calcul de $(C^{B,t+1}, \partial^{B,t+1})$ par application du théorème SECE à l'identification ou à la désidentification implique l'ajout de nouveaux générateurs à $(C^{B,t}, \partial^{B,t})$. Nous montrons [3] que les applications successives du théorème SECE :

— induisent une filtration sur les complexes de chaînes intermédiaires dans le cas de l'identification ;

— n'induisent pas de filtration sur les complexes de chaînes intermédiaires dans le cas de la désidentification.

Démonstration. Présentons la notion de filtration sur des complexes de chaînes [18, p.814], définie jusqu'ici uniquement sur les complexes simpliciaux (cf. définition 1.4.2).

Définition 4.2.1. *Soit (C, ∂) un complexe de chaînes. Une filtration de complexes de chaînes est une séquence :*

$$\emptyset = C_0 \subseteq C_1 \subseteq \ldots \subseteq C_n = C$$

telle que $C_i\partial \subseteq Ci$, pour tout i compris entre 0 et n.

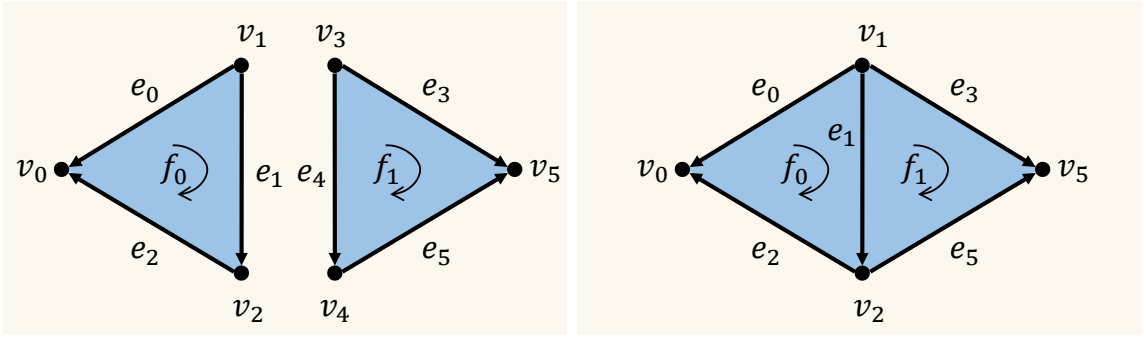
Soient ST une structure topologique et ST' la structure résultant de l'application d'une identification sur ST. Nous avons vu (cf. paragraphe 1.4.4-b) que ST n'est pas incluse dans ST' et que ST' n'est pas incluse dans ST (cf. figure 1.42). Comme l'illustre la figure 4.1, il en est de même pour les complexes de chaînes associés à ces structures topologiques. Soient (C, ∂) le complexe de chaînes associé à ST et (C', ∂') le complexe de chaînes associé à ST', on a :

1. Pour rappel, l'homologie persistante est présentée dans la sous-section 1.4.2.

2. Parfois, $(C^B, \partial^B) = (C, \partial)$ à l'étape 0 d'un processus de construction, puis, à chaque étape, de nouveaux générateurs lui sont ajoutés.

3. Nous remercions Mathijs WINTRAECKEN pour sa remarque lors des JGA2019, à l'origine du questionnement autour de l'existence d'une filtration au niveau des complexes de chaînes intermédiaires.

— $C \not\subseteq C'$, donc (C, ∂) n'est pas inclus dans (C', ∂') ;
— $C'\partial \not\subseteq C'$, donc (C', ∂') n'est pas inclus dans (C, ∂).



(a) (C, ∂) (b) (C', ∂')

FIGURE 4.1 – Représentation graphique de deux complexes de chaînes (C, ∂) et (C', ∂') associés aux ensembles semi-simpliciaux de la figure 1.8. Selon le sens de lecture, l'un résulte d'une identification ou d'une désidentification appliquée à l'autre. L'identification consiste à identifier v_1 avec v_3, v_2 avec v_4 et e_1 avec e_4. Dans le sens de l'identification, c'est-à-dire si (C', ∂') succède à (C, ∂), il n'y a pas de filtration car $C \not\subseteq C'$. Plus précisément, $e_4 \in C$ et $e_4 \notin C'$. Dans l'autre sens, il n'y a pas non plus de filtration car $C'\partial \not\subseteq C'$. Plus précisément, $f_1\partial = -e_4 + e_3 - e_5$ et $e_4 \notin C'$.

Intéressons-nous à la séquence de complexes de chaînes intermédiaires $(C^{B,t}, \partial^{B,t})$. Rappelons que les formules de calcul de $\partial^{B,t+1}$ dans l'application du théorème SECE à l'identification et à la désidentification sont, respectivement :

$$\partial^{B,t+1} = \begin{pmatrix} -\partial^B & i^B \\ 0 & \partial^{B,t} \end{pmatrix} \qquad \partial^{B,t+1} = \begin{pmatrix} \partial^{B,t} & \chi^B \\ 0 & \partial^B \end{pmatrix}$$

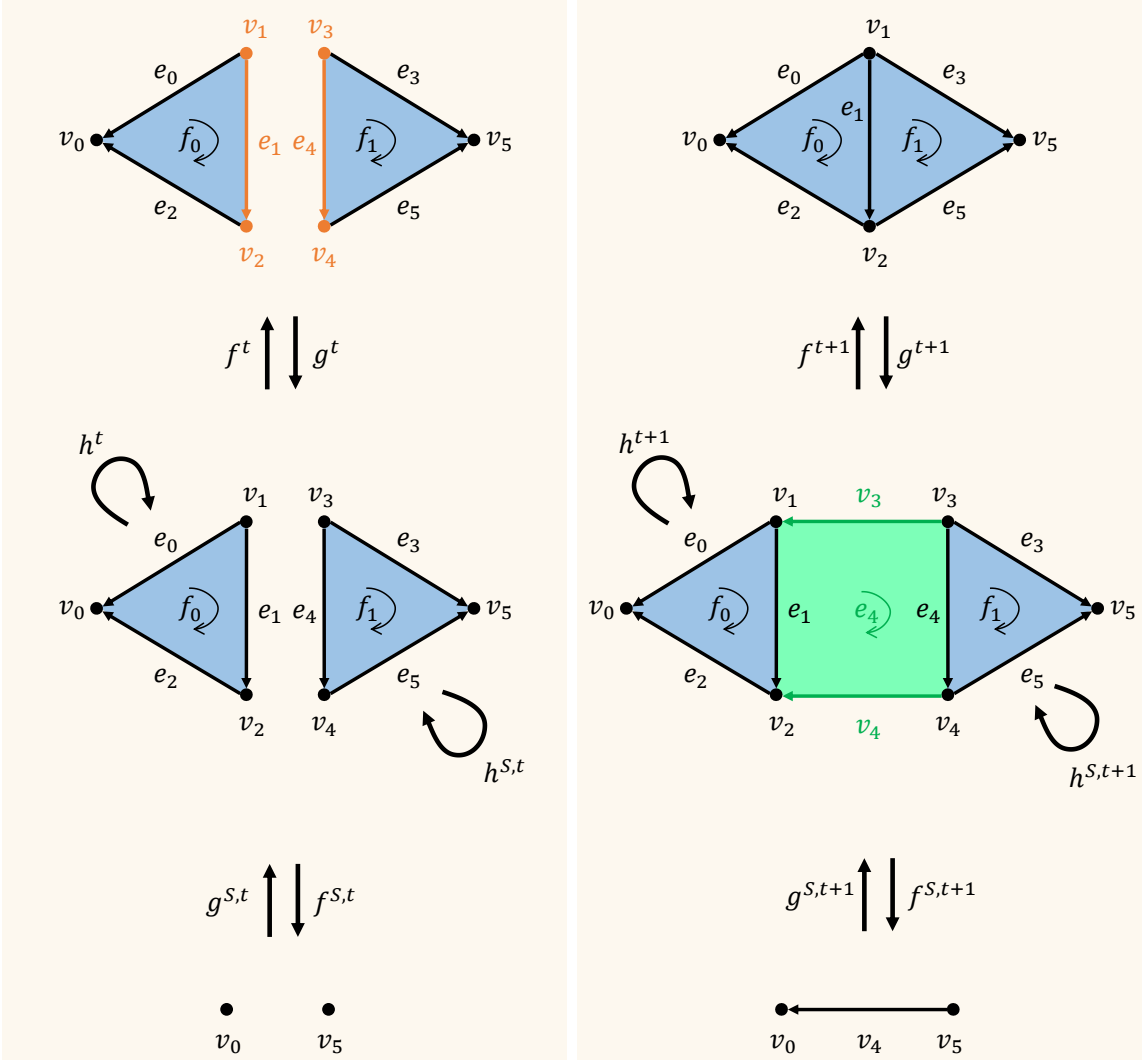
Comme dit précédemment, dans les deux cas, le calcul de $(C^{B,t+1}, \partial^{B,t+1})$ implique l'ajout de nouveaux générateurs dans $(C^{B,t}, \partial^{B,t})$. De fait, $C^{B,t} \subseteq C^{B,t+1}$. Intéressons-nous aux opérateurs de bord :

— dans le cas de l'identification, on constate que le bord des générateurs de C^t n'est pas modifié dans le calcul de $(C^{t+1}, \partial^{t+1})$ du fait de la ligne $\begin{pmatrix} 0 & \partial^{B,t} \end{pmatrix}$. En conséquence, $C^{B,t} \subseteq C^{B,t+1}$ et $C^{B,t}\partial^{B,t+1} \subseteq C^{B,t}$. Dit autrement, $(C^{B,t}, \partial^{B,t})$ est inclus dans $(C^{B,t+1}, \partial^{B,t+1})$, et induit une filtration ;
— dans le cas de la désidentification, le bord des générateurs de C^t est modifié dans le calcul de $(C^{t+1}, \partial^{B,t+1})$ du fait de la ligne $\begin{pmatrix} \partial^{B,t} & \chi^B \end{pmatrix}$. En conséquence, $C^{B,t}\partial^{B,t+1} \not\subseteq C^{B,t}$. Dit autrement, $(C^{B,t}, \partial^{B,t})$ n'est pas inclus dans $(C^{B,t+1}, \partial^{B,t+1})$, et n'induit pas de filtration.

Les figures 4.2 et 4.3 illustrent ces observations sur des équivalences homologiques résultant de l'application du théorème SECE à une identification et à une désidentification.

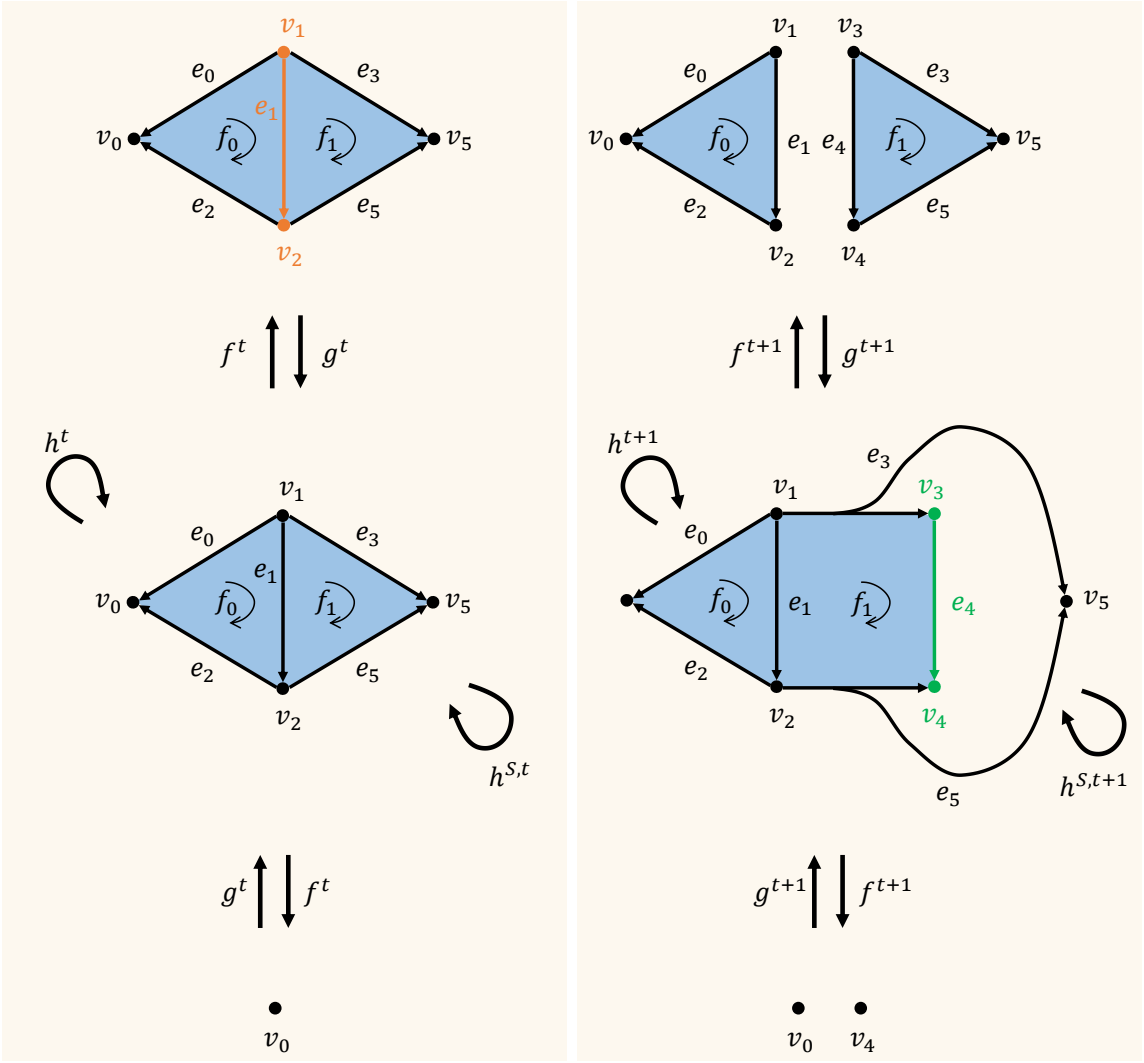
Remarque. Plus généralement, toute séquence d'opération relevant du cas 1 du théorème SECE induit une filtration sur les complexes de chaînes intermédiaires, ce qui n'est pas le cas si une opération relève du cas 2. En particulier, comme nous le verrons dans le paragraphe 4.3.4-a, la dégénérescence relève du cas 1.

Conclusion. On constate donc que l'application du théorème SECE à l'identification induit une filtration au niveau des complexes de chaînes intermédiaires de l'équivalence homologique maintenue et que son application à la désidentification n'induit pas de filtration. Pour autant, rappelons que la désidentification est l'inverse de l'identification, et peut être traitée comme telle. Plus précisément, cela veut dire qu'en distinguant deux sens de progression dans un processus de construction, de t vers $t+1$, le sens "normal" (ou zig), et de $t+1$ vers t, le sens "inverse" (ou zag), on peut utiliser, à chaque étape, l'application du théorème SECE à l'identification de manière à ce que le processus de construction induise une filtration zigzag. Ceci permet d'établir une correspondance entre l'homologie persistante "de base" et le théorème SECE, et peut servir de point de départ au transfert de résultat de l'une des méthodes à l'autre.



(a) γ^t (b) γ^{t+1}

FIGURE 4.2 – La mise à jour d'une équivalence homologique suite à l'application du théorème SECE sur l'identification des générateurs en orange (extrait de la figure 1.43). En vert, les cellules qui ont été créées dans $(C^{B,t+1}, \partial^{B,t+1})$ par rapport à $(C^{B,t}, \partial^{B,t})$. (a) L'équivalence homologique avant identification. (b) L'équivalence homologique après identification.



(a) γ^t (b) γ^{t+1}

FIGURE 4.3 – La mise à jour d'une équivalence homologique suite à l'application du théorème SECE à la désidentification des générateurs en orange (extrait de la figure 1.44). En vert, les cellules qui ont été créées dans $(C^{B,t+1}, \partial^{B,t+1})$ par rapport à $(C^{B,t}, \partial^{B,t})$. (a) L'équivalence homologique avant désidentification. (b) L'équivalence homologique après désidentification.

4.3 Homologie persistante de tours

Les documents de référence pour cette section sont [2] et [19]. Dans ces documents, la persistance de tours est présentée pour les complexes simpliciaux (cf. paragraphe 1.2.1-a).

4.3.1 Préambule

Le fait que l'homologie persistante "de base" repose sur la notion de filtration est une limitation bloquante pour l'appliquer à notre cadre d'étude. En effet, comme nous l'avons vu dans la section 4.2, il n'est pas toujours possible d'associer une filtration à une séquence quelconque d'identifications et de désidentifications. L'homologie persistante "de tours" lève cette limitation en utilisant deux résultats importants :

1. toute application simpliciale se décompose en une suite *d'inclusions et d'effondrements élémentaires* ;
2. une suite d'*inclusions* est une filtration et relève de l'homologie persistante "de base". *L'effondrement élémentaire* peut être "décomposé" en *inclusions*.

Objectifs. De ce fait, on s'interroge sur l'existence d'une éventuelle "passerelle" faisant le lien entre l'application du théorème SECE à l'identification et à la désidentification, et la persistance de tours. À notre connaissance, la persistance de tours n'est définie que sur les complexes simpliciaux. L'identification et la désidentification sont des opérations définies sur d'autres structures topologiques. Pour établir un lien entre l'application du théorème SECE à ces opérations et la persistance de tours, il convient donc d'abord de toutes les exprimer dans un même cadre.

L'objectif de cette section est de poser ce cadre, puis de dégager des pistes de comparaison pouvant servir de point de départ pour une étude plus approfondie des deux méthodes. Cette dernière ne fait pas partie de ce document.

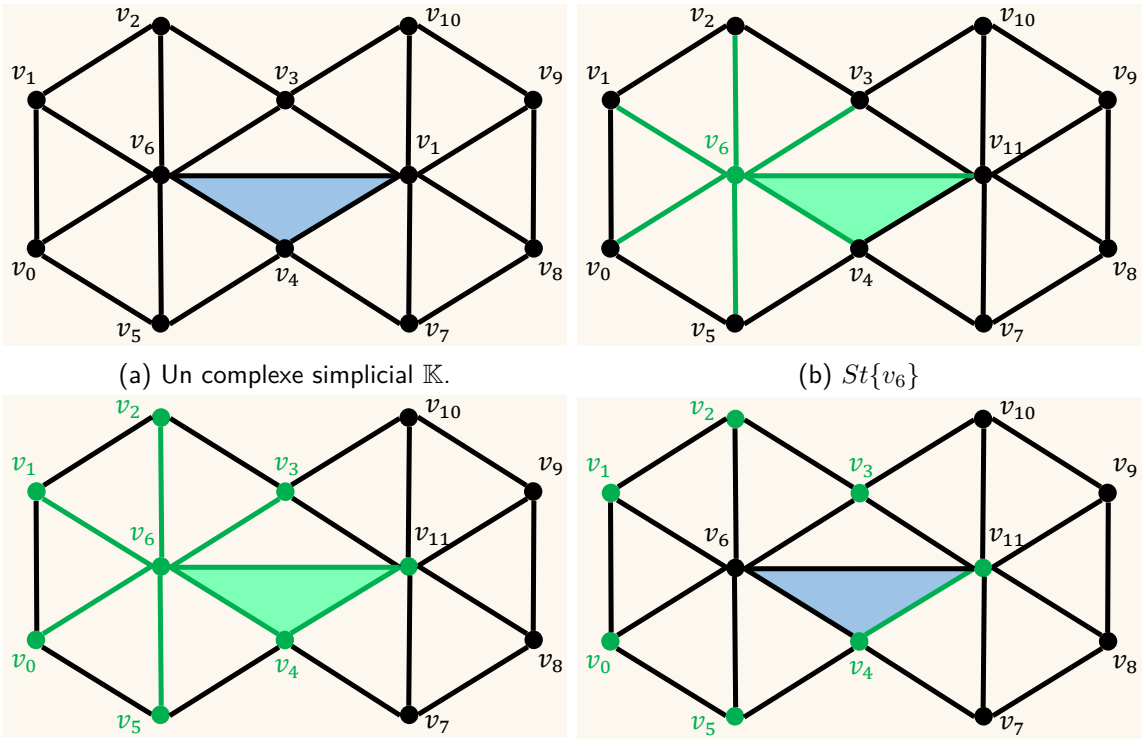
4.3.2 Notions préliminaires

Définition 4.3.1. *Une **tour** de taille m est une collection de complexes simpliciaux $\mathbb{K}_0, \dots \mathbb{K}_m$ et d'applications simpliciales $\phi_0 \dots \phi_{m-1}$. En particulier, pour i tel que $0 \leq i \leq m-1$:*

- *une tour est **monotone** si toutes les applications simpliciales qui la composent sont de la forme $\phi_i : \mathbb{K}_i \to \mathbb{K}_{i+1}$;*
- *une tour est **zigzag** si les applications simpliciales qui la composent sont de la forme $\phi_i : \mathbb{K}_i \to \mathbb{K}_{i+1}$ ou $\phi_i : \mathbb{K}_i \leftarrow \mathbb{K}_{i+1}$.*

Définition 4.3.2. *Soit $\mathbb{X}$ un sous-ensemble d'un complexe simplicial $\mathbb{K}$. L'**étoile** de $\mathbb{X}$, notée $St\mathbb{X}$, est l'ensemble $St\mathbb{X} = \{\sigma' \in \mathbb{K} | \exists \sigma \in \mathbb{X}, \sigma \subseteq \sigma'\}$. La **fermeture** de $\mathbb{X}$, notée $\overline{\mathbb{X}}$, est le complexe simplicial formé par $\mathbb{X}$ et l'ensemble de ses faces. Le **lien** de $\mathbb{X}$, noté $Lk\mathbb{X}$, est l'ensemble $Lk\mathbb{X} = \overline{St\mathbb{X}} \setminus St\overline{\mathbb{X}}$.*

La définition 4.3.2 est illustrée sur la figure 4.4.



(a) Un complexe simplicial $\mathbb{K}$.

(b) $St\{v_6\}$

(c) $\overline{St\{v_6\}}$

(d) $Lk\{v_6\}$

FIGURE 4.4 – L'étoile, la fermeture de l'étoile et le lien pour $\mathbb{X} = \{v_6\}$. Les simplexes concernés par chaque notion sont mis en évidence en vert.

Définition 4.3.3. *Soient $\mathbb{K}$ et $\mathbb{K}'$ deux complexes simpliciaux. Une **inclusion élémentaire** $\phi : \mathbb{K} \to \mathbb{K}'$ est une application simpliciale injective telle que K' a au plus un simplexe de plus que $\mathbb{K}$.*

Définition 4.3.4. *Soient $\mathbb{K}$ et $\mathbb{K}'$ deux complexes simpliciaux. Un **effondrement élémentaire** $\phi : \mathbb{K} \to \mathbb{K}'$ est une application simpliciale surjective. Elle est injective pour tous les sommets sauf pour un couple de sommets (u, v), qu'elle associe à un même sommet.*

Les définitions 4.3.3 et 4.3.4 sont illustrées sur les figures 4.5 et 4.6.

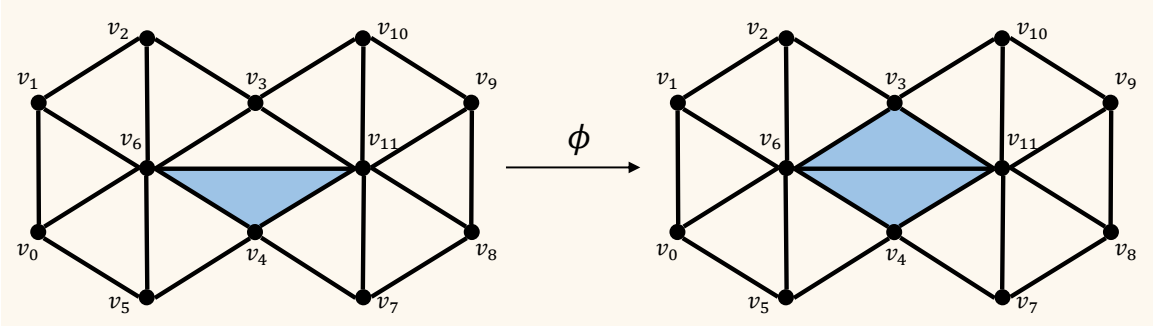


FIGURE 4.5 – Une inclusion élémentaire $\phi : \mathbb{K} \to \mathbb{K}'$. Le simplexe de $K' \setminus K$ est $\{v_3, v_6, v_{11}\}$.

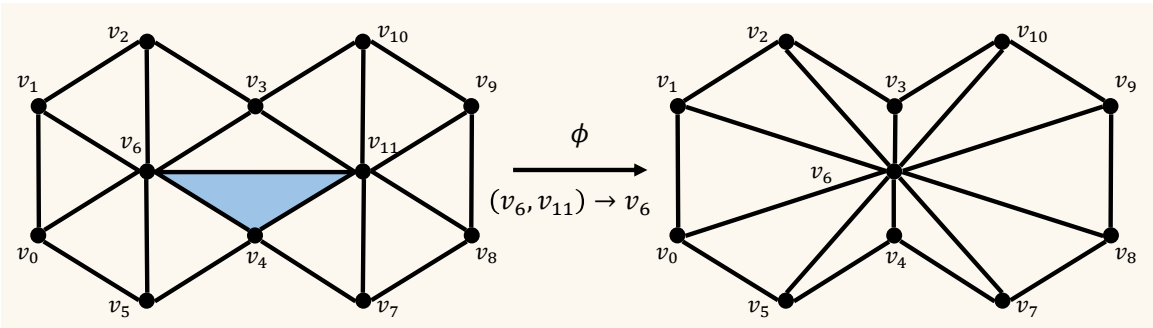


FIGURE 4.6 – Un effondrement élémentaire $\phi : \mathbb{K} \to \mathbb{K}'$. En particulier, $v_6\phi = v_{11}\phi = v_6$.

4.3.3 Fonctionnement

L'homologie persistante de tours est une extension de l'homologie persistante "de base", où les applications simpliciales étudiées ne sont pas nécessairement des inclusions. D'après la proposition 2.5 de [2], toute application simpliciale se décompose en une séquence d'inclusions ou d'effondrements élémentaires. L'inclusion élémentaire relève de l'homologie persistante "de base". L'effondrement élémentaire fait l'objet des travaux de [2]. Les auteurs distinguent deux cas :

— le cas général des tours *zigzags* ;

— le cas particulier des tours monotones.

4.3.3-a Cas général : tours zigzags

Pour traiter l'effondrement élémentaire dans une tour zigzag, les auteurs de [2] proposent d'utiliser l'opération de cône des complexes simpliciaux.

Définition 4.3.5. *Soit $\mathbb{K}$ un complexe simplicial et $\mathbb{X}$ un sous-complexe de $\mathbb{K}$. Le **cône** d'un sommet $u \in \mathbb{K}$ avec $\mathbb{X}$, noté $u * \mathbb{X}$, est le complexe simplicial défini par :*

$$u * \mathbb{X} = \{\overline{\sigma \cup \{u\}}\} \quad \forall \sigma \in \mathbb{X}$$

Soit un effondrement élémentaire $\phi : \mathbb{K} \to \mathbb{K}'$ associant les sommets (u, v) de $\mathbb{K}$ au sommet u de $\mathbb{K}'$. Considérons le complexe simplicial $\widehat{\mathbb{K}}$ défini par :

$$\widehat{\mathbb{K}} = \mathbb{K} \cup (u * \overline{St\{v\}})$$

D'après la propriété 2.6 de [2], ce dernier est tel que :

— $\mathbb{K} \subseteq \widehat{\mathbb{K}}$;

— $\mathbb{K}' \subseteq \widehat{\mathbb{K}}$.

Dit autrement, les complexes simpliciaux $\mathbb{K}$, $\mathbb{K}'$ et $\widehat{\mathbb{K}}$ sont reliés par les relations d'inclusion $i : \mathbb{K} \to \widehat{\mathbb{K}}$ et $i' : \mathbb{K}' \to \widehat{\mathbb{K}}$. En particulier, d'après la proposition 2.7 de [2], i' induit un isomorphisme des groupes d'homologie de $\mathbb{K}'$ sur ceux de $\widehat{\mathbb{K}}$, et les groupes d'homologie de $\mathbb{K}$ peuvent être associés à ceux de $\mathbb{K}'$ via la relation $i(i')^{-1}$.

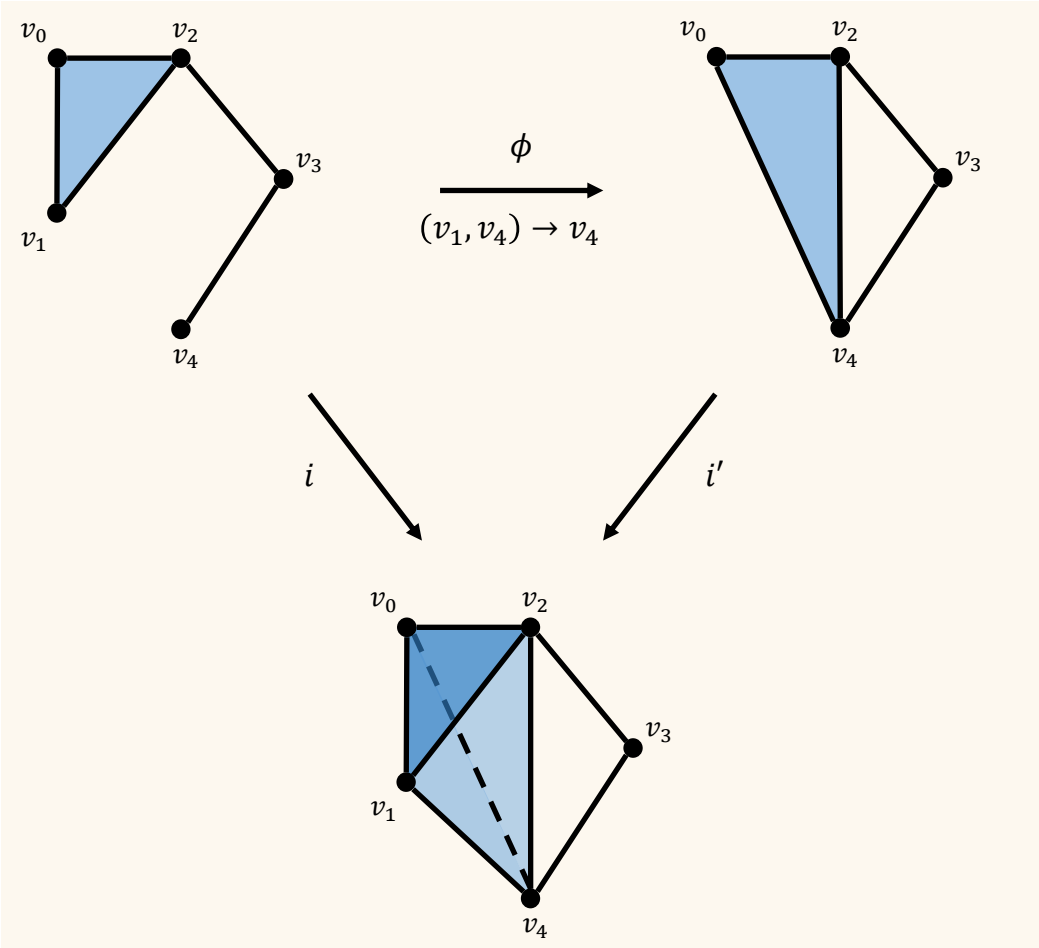


FIGURE 4.7 – Un complexe simplicial $\mathbb{K}$ auquel est appliqué un effondrement élémentaire $\phi : \mathbb{K} \to \mathbb{K}'$ associant les sommets (v_1, v_4) de $\mathbb{K}$ au sommet v_4 de $\mathbb{K}'$. En bas, le complexe simplicial $\widehat{\mathbb{K}} = \mathbb{K} \cup (v_4 * \overline{St\{v_1\}})$ qui induit les relations d'inclusion $i : \mathbb{K} \to \widehat{\mathbb{K}}$ et $i' : \mathbb{K} \to \widehat{\mathbb{K}}$. Cette figure s'inspire de la figure 1 de [2].

Exemple. Soit un complexe simplicial

$$\mathbb{K} = \{\{v_0\}, \{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_0, v_1\}, \{v_0, v_2\}, \{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\}, \{v_0, v_1, v_2\}\}$$

et un effondrement élémentaire $\phi : \mathbb{K} \to \mathbb{K}'$ associant le couple (v_1, v_4) de $\mathbb{K}$ au sommet v_4 de $\mathbb{K}'$. Le complexe simplicial $\mathbb{K}'$ est :

$$\mathbb{K}' = \{\{v_0\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_0, v_4\}, \{v_0, v_2\}, \{v_4, v_2\}, \{v_2, v_3\}, \{v_3, v_4\}, \{v_0, v_4, v_2\}\}$$

En particulier :

— l'étoile de v_1 est $St\{v_1\} = \{\{v_1\}, \{v_0, v_1\}, \{v_1, v_2\}, \{v_0, v_1, v_2\}\}$;

— la fermeture de $St\{v_1\}$ est $\overline{St\{v_1\}} = St\{v_1\} \cup \{\{v_0\}, \{v_2\}, \{v_0, v_2\}\}$.

Le complexe simplicial $\widehat{\mathbb{K}} = \mathbb{K} \cup (v_4 * \overline{St\{v_1\}})$ est alors :

$$\widehat{\mathbb{K}} = \mathbb{K} \cup \{v_1, v_4\}, \{v_0, v_1, v_4\}, \{v_1, v_2, v_4\}, \{v_0, v_1, v_2, v_4\}, \{v_0, v_4\}, \{v_2, v_4\}, \{v_0, v_2, v_4\}\}$$

On constate que $\mathbb{K} \subseteq \widehat{\mathbb{K}}$ et $\mathbb{K}' \subseteq \widehat{\mathbb{K}}$. Cet exemple est illustré sur la figure 4.7.

4.3.3-b Cas particulier : tours monotones

Pour traiter l'effondrement élémentaire dans le cas des tours monotones, les auteurs de [2] proposent d'utiliser un algorithme basé sur la notion d'annotations. Plus précisément, l'objectif est de maintenir les annotations d'un complexe simplicial annoté au fil d'inclusions et d'effondrements élémentaires.

Définition 4.3.6. *Soit $\mathbb{K}$ un complexe simplicial. Notons $\mathbb{K}^p$ l'ensemble des p-simplexes de $\mathbb{K}$. Une **annotation** $a : \mathbb{K}^p \to \mathbb{Z}_2^g$ est une affectation d'un mot binaire $a_\sigma = \sigma a$ de longueur g à chaque p-simplexe σ de $\mathbb{K}^p$:*

- *g est le rang du p-ème groupe d'homologie à coefficients dans $\mathbb{Z}_2$ de $\mathbb{K}$;*
- *deux p-cycles z_0 et z_1 ont le même mot binaire si et seulement s'ils appartiennent à une même classe d'homologie (à coefficients dans $\mathbb{Z}_2$).*

L'annotation d'une p-chaîne c_p est $a_{c_p} = \sum_{\sigma \in c_p} a_\sigma$.

Remarque. La définition 4.3.6 est plus restrictive que la définition d'une annotation donnée dans [2]. Elle condense la notion d'annotation et d'annotation valide. L'annotation du complexe simplicial initial est une donnée d'entrée. Un algorithme de calcul d'une annotation est donné dans [81]. De plus, dans [2], les bits des annotations sont estampillés avec une valeur entière qui correspond à l'étape de construction à laquelle la classe d'homologie représentée par un bit est créée. Cette valeur entière permet par exemple de calculer un diagramme de persistance. L'objectif du paragraphe étant simplement de présenter la façon dont les annotations sont maintenues au fil des opérations, nous omettons cette valeur entière.

Inclusion élémentaire. Soit $\mathbb{K}$ un complexe simplicial annoté et $\phi : \mathbb{K} \to \mathbb{K}'$ une inclusion élémentaire telle que $K' = K \cup \{\sigma\}$. Les annotations des simplexes de $\mathbb{K}'$ se calculent à partir de $a_{\sigma\partial}$, c'est-à-dire de l'annotation du bord de σ dans $\mathbb{K}$. Deux cas se distinguent en fonction de $a_{\sigma\partial}$:

1. $a_{\sigma\partial}$ est nulle. Dans ce cas,
   - on augmente de 1 la taille du mot binaire associé à tout p-simplexe de $\mathbb{K}$. Soit $[b_0, b_1, \ldots, b_{g-1}]$ le mot binaire d'un p-simplexe de $\mathbb{K}$. Ce simplexe est annoté $[b_0, b_1, \ldots, b_{g-1}, 0]$ dans $\mathbb{K}'$;
   - le simplexe σ est annoté $[0 \ldots 0, 1]$ dans $\mathbb{K}'$;
2. $a_{\sigma\partial}$ n'est pas nulle. Dans ce cas, on annule le u-ème bit de l'annotation de tous les $(p-1)$-simplexes, où u désigne l'indice du dernier bit non nul de $a_{\sigma\partial}$. Pour ce faire :
   - on ajoute $a_{\sigma\partial}$ à toutes les annotations des $(p-1)$-simplexes dont l'annotation est telle que $b_u = 1$;
   - on réduit de 1 la taille du mot binaire associé à tout $(p-1)$-simplexe en supprimant b_u de leur annotation.

   Notons que si $a_{\sigma\partial}$ n'est pas nulle, σ n'est, par définition, pas un p-cycle (cf. définition 1.3.6). Son annotation est nulle et de la taille des annotations des p-simplexes.

Remarque. La justification de ces calculs fait l'objet de la partie 5 de [2]. Intuitivement, l'ajout ou la suppression d'un bit des annotations des p-simplexes représente respectivement la création ou la suppression d'une classe d'homologie du p-ème groupe.

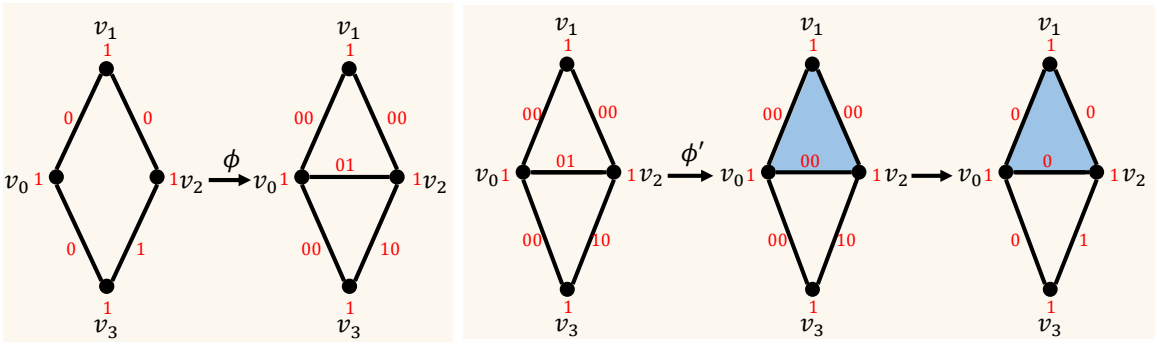


FIGURE 4.8 – Mise à jour des annotations d'un complexe simplicial lors d'une inclusion élémentaire. En rouge, les annotations des simplexes. (a) Une inclusion élémentaire $\phi : \mathbb{K} \to \mathbb{K}'$ telle que $\mathbb{K}' \setminus \mathbb{K} = \{v_0, v_2\}$. En particulier, $a_{\{v_0,v_2\}\partial} = 1+1 = 0$, donc il s'agit du cas 1 de l'inclusion élémentaire. (b) Une inclusion élémentaire $\phi' : \mathbb{K}' \to \mathbb{K}''$ telle que $\mathbb{K}' \setminus \mathbb{K} = \{v_0, v_1, v_2\}$. En particulier, $a_{\{v_0,v_1,v_2\}\partial} = 00 + 00 + 01 = 01$, donc il s'agit du cas 2 de l'inclusion élémentaire. L'annotation de $\{v_0, v_1, v_2\}$ est vide car le rang de H^2 est 0.

Exemple. La figure 4.8a illustre le cas 1 de l'inclusion élémentaire car $a_{\{v_0,v_2\}\partial} = 0$. La mise à jour des annotations consiste donc à ajouter un bit au mot binaire de chaque 1-simplexe. En particulier :

- l'annotation de $\{v_0, v_1\}, \{v_1, v_2\}, \{v_0, v_3\}$ devient 00 ;
- l'annotation de $\{v_2, v_3\}$ devient 10 ;
- l'annotation de $\{v_0, v_2\}$ est 01.

La figure 4.8b illustre le cas 2 de l'inclusion élémentaire car $a_{\{v_0,v_1,v_2\}\partial} = 01$. La mise à jour des annotations consiste donc à :

- ajouter 01 à toutes les annotations des 1-simplexes telles que $b_1 = 1$. En l'occurrence, seule l'arête $\{v_0, v_1\}$ est concernée, son annotation devient $01 + 01 = 00$;
- on supprime b_1 de toutes les annotations des 1-simplexes, c'est-à-dire que l'annotation des simplexes $\{v_0, v_1\}, \{v_1, v_2\}, \{v_0, v_2\}$ et $\{v_0, v_3\}$ devient 0, et celle de $\{v_2, v_3\}$ devient 1.

Effondrement élémentaire. La mise à jour des annotations d'un complexe simplicial $\mathbb{K}$ auquel est appliqué un effondrement élémentaire $\phi : \mathbb{K} \to \mathbb{K}'$ se décompose en 2 étapes :

1. l'inclusion élémentaire de simplexes pour satisfaire la *condition de lien* ;
2. le "transfert" d'annotations, qui consiste à "préparer" les annotations pour qu'elles ne soient pas obsolètes après l'effondrement élémentaire ;

Condition de lien. La première étape consiste à insérer des simplexes dans $\mathbb{K}$ pour garantir que la suppression de simplexes induite par l'effondrement ne nécessite pas de mise à jour des annotations. Dit autrement, on fait en sorte que les annotations de $\mathbb{K}$ soient "prêtes" pour l'effondrement. Pour cela, on insère des simplexes dans $\mathbb{K}$ de manière à ce que l'effondrement vérifie la *condition de lien*.

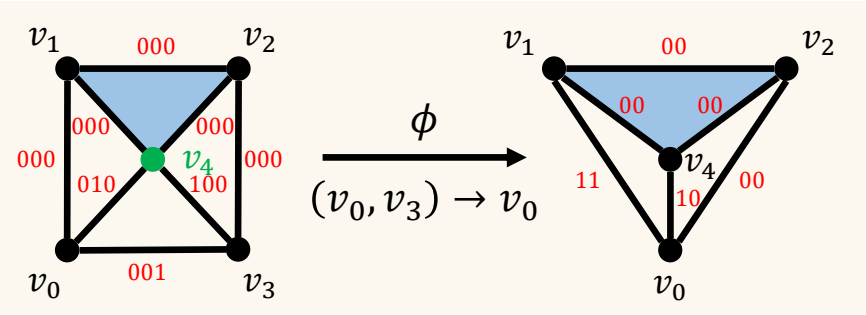


FIGURE 4.9 – Un effondrement élémentaire $\phi : \mathbb{K} \to \mathbb{K}'$ tel que $v_0\phi = v_3\phi = v_0$. En vert, $Lk\{v_0\} \cap Lk\{v_3\} = \{v_4\}$. En rouge, les annotations des 1-simplexes.

Définition 4.3.7. *Soit u, v deux sommets d'un complexe simplicial $\mathbb{K}$. Ils vérifient la* ***condition de lien*** *si $Lk\{u\} \cap Lk\{v\} = Lk\{u, v\}$. Plus généralement, un effondrement élémentaire $\phi : \mathbb{K} \to \mathbb{K}'$ vérifie la condition de lien si le couple de sommets pour lequel il est non injectif vérifie la condition de lien.*

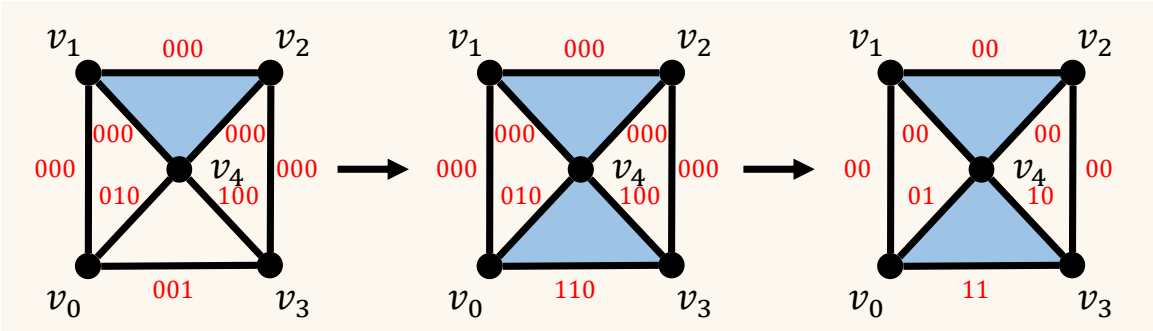


FIGURE 4.10 – L'inclusion élémentaire du 2-simplexe $\{v_0, v_3, v_4\}$ à l'issue de laquelle la condition de lien est vérifiée sur v_0 et v_3, c'est-à-dire que $Lk\{v_0\} \cap Lk\{v_3\} = Lk\{v_0, v_3\} = \{v_4\}$. En rouge, les annotations des 1-simplexes. En particulier, notons que l'annotation de $\{v_0, v_3\}$ devient d'abord 110 car $a_{\{v_0,v_3\}} + a_{\{v_0,v_3,v_4\}\partial} = 001 + 001 + 010 + 100$, puis 11 car on supprime des annotations des 1-simplexes les bits aux indices du dernier bit non nul de $a_{\{v_0,v_3,v_4\}\partial}$, c'est-à-dire b_2.

Exemple. Considérons l'effondrement élémentaire $\phi : \mathbb{K} \to \mathbb{K}'$ illustré sur la figure 4.9, associant les sommets v_0 et v_3 de $\mathbb{K}$ au sommet v_0 de $\mathbb{K}'$. Observons que :

— $Lk\{v_0\} \cap Lk\{v_3\} = \{v_4\}$;

— $Lk\{v_0, v_3\} = \emptyset$.

La condition de lien n'est donc pas vérifiée par v_0 et v_3 car $Lk\{v_0\} \cap Lk\{v_3\} \neq Lk\{v_0, v_3\}$. Pour maintenir les annotations des simplexes de $\mathbb{K}$, il faut donc commencer par insérer des simplexes de manière à ce que la condition de lien soit vérifiée. En l'occurrence, le seul simplexe à insérer est $\{v_0, v_3, v_4\}$; cette opération est une inclusion élémentaire[4]. Plus précisément, les annotations sont mises à jour en fonction du cas 2 de l'inclusion élémentaire car $a_{\{v_0,v_3,v_4\}\partial} = 010 + 100 + 001 = 111$. En particulier :

4. Aucun algorithme n'est donné dans [2] pour déterminer les simplexes à insérer.

— l'annotation de $\{v_0, v_3\}$ devient 110 car elle est la seule dont le bit $b_2 = 1$;

— le bit b_2 est supprimé de toutes les annotations des 1-simplexes.

À l'issue de cette mise à jour, notons que :

— $Lk\{v_0\} \cap Lk\{v_3\} = \{v_4\}$;

— $Lk\{v_0, v_3\} = \{v_4\}$.

La condition de lien est donc vérifiée car $Lk\{v_0\} \cap Lk\{v_3\} = Lk\{v_0, v_3\}$. Pour finir de traiter ϕ, il faut ensuite procéder au transfert d'annotations.

Transfert d'annotations. Cette opération distingue deux types de simplexes : les *simplexes fuyants* et les *simplexes miroirs*.

Définition 4.3.8. *Soit un effondrement élémentaire $\phi : \mathbb{K} \to \mathbb{K}'$ non injectif pour les sommets u et v. Un simplexe σ de $\mathbb{K}$ est dit **fuyant** si $|\sigma\phi| = |\sigma| - 1$. Deux simplexes σ et σ' de $\mathbb{K}$ sont dits **miroirs** s'ils ne divergent que d'un sommet, et que ces sommets sont u et v.*

Le transfert d'annotation a pour but de permettre l'application de l'effondrement élémentaire en conservant les annotations déjà en place. Dit autrement, la baisse de dimension des simplexes fuyants et la suppression d'une partie des simplexes miroirs n'entraîne aucune perte d'information homologique représentée par les annotations : elles sont "prêtes" pour l'effondrement. Pour cela, il faut que :

— les annotations des simplexes fuyants soient nulles ;

— les annotations des simplexes miroirs soient les mêmes.

Observons que :

— les simplexes fuyants sont tous les simplexes σ de $\mathbb{K}$ tels que $\{u, v\} \subseteq \sigma$;

— exactement 2 faces d'un p-simplexe fuyant sont des $(p-1)$-simplexes miroirs.

Pour faire en sorte que les annotations soient "prêtes" pour l'effondrement, il faut *transférer les annotations* de tous les simplexes fuyants.

Définition 4.3.9. *Soient un effondrement élémentaire $\phi : \mathbb{K} \to \mathbb{K}'$ associant les sommets u et v à u, et τ un p-simplexe fuyant et σ sa $(p-1)$-face miroir composée de u. Le **transfert de l'annotation** de τ consiste à ajouter a_τ à son annotation ainsi qu'à celles de son étoile directe*[5].

À l'issue du transfert des annotations de tous les simplexes fuyants, l'effondrement élémentaire est appliqué sur $\mathbb{K}$.

Exemple. La figure 4.11 illustre le transfert d'annotations des 1-simplexes du complexe simplicial qui résulte de l'inclusion élémentaire illustrée sur la figure 4.10. Considérons les 1-simplexes :

— $\{v_0, v_3\}$ est le seul 1-simplexe fuyant. Son annotation $a_{\{v_0,v_3\}}$ est donc la seule transférée ;

— sa face miroir composée de v_0 est $\{v_0\}$, dont les cofaces de codimension 1 sont $\{v_0, v_1\}$, $\{v_0, v_4\}$, $\{v_0, v_3\}$;

— l'annotation $a_{\{v_0,v_3\}} = 11$ est ajoutée aux annotations de ces cofaces, c'est-à-dire que

  · $a_{\{v_0,v_1\}} = 00 + 11 = 11$;

  · $a_{\{v_0,v_4\}} = 01 + 11 = 10$;

5. Pour rappel, l'étoile directe d'un simplexe est son étoile restreinte aux $p+1$-simplexes.

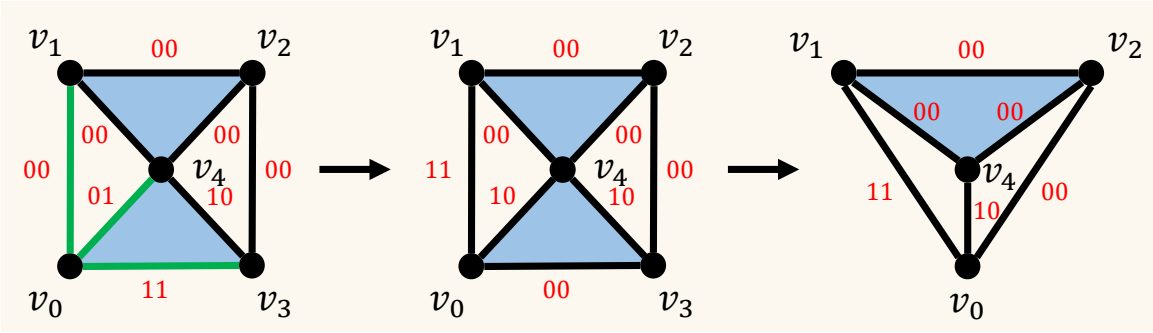


FIGURE 4.11 – Le transfert d'annotations préalable à l'application de l'effondrement élémentaire ϕ. Les simplexes fuyants sont $\{v_0, v_3\}$ et $\{v_0, v_3, v_4\}$. En rouge, les annotations des 1-simplexes. En vert, les 1-simplexes cofaces de codimension 1 de $\{v_0\}$ auxquels l'annotation de $\{v_0, v_3\}$ est ajoutée.

· $a_{\{v_0,v_3\}} = 11 + 11 = 00.$

En particulier, notons qu'à l'issue du transfert de $a_{\{v_0,v_3\}}$:

— le 1-simplexe fuyant $\{v_0, v_3\}$ a une annotation nulle ;

— les 1-simplexes miroirs $\{v_0, v_4\}$ et $\{v_3, v_4\}$ ont la même annotation 10.

À ce stade, il ne reste plus qu'à appliquer l'effondrement élémentaire : les annotations des simplexes sont valides, et peuvent être utilisées comme point de départ d'une autre application simpliciale.

4.3.4 Cadre commun

Nous avons vu que la persistance de tours traite les applications simpliciales en les décomposant en deux opérations élémentaires : l'inclusion et l'effondrement. Comme nous l'avons vu jusqu'ici, le théorème SECE est applicable à l'identification et à la désidentification. Nous montrons dans le paragraphe 4.3.4-a qu'il s'applique également à la dégénérescence. Dans le paragraphe 4.3.4-b, nous établissons un lien entre la persistance de tours et les applications du théorème SECE en montrant qu'un effondrement élémentaire appliqué à un complexe simplicial $\mathbb{K}$ induit des identifications et/ou des dégénérescences sur l'ensemble simplicial (K, d, s) associé à $\mathbb{K}$. Dans le paragraphe 4.3.4-c, nous donnons un exemple de passage d'un cadre à l'autre.

4.3.4-a Application du théorème SECE à la dégénérescence

Pour rappel, nous présentons la dégénérescence dans le paragraphe 1.2.1-c, sur les ensembles simpliciaux. Nous montrons qu'une dégénérescence induit une SECE et entre donc dans le cadre du théorème SECE.

Soit un complexe de chaînes (C^t, ∂^t) induit par un ensemble simplicial (K, d, s) [6] dans lequel on dégénère des simplexes de bord nul dans (K, d, s). Soit σ un simplexe dégénéré, le générateur qu'il induit dans (C^t, ∂^t) vérifie $\sigma\partial^t = 0_{C^t}$. Nommons $(C^{t+1}, \partial^{t+1})$ le complexe de chaînes induit par (K, d, s) après dégénérescence. La SECE S,

$$(C, \partial) \underset{\text{r}}{\overset{\text{i}}{\leftrightarrows}} (C^t, \partial^t) \underset{\text{s}}{\overset{\text{j}}{\leftrightarrows}} (C^{t+1}, \partial^{t+1})$$

induite par la dégénérescence est définie par :

- C est engendré par les simplexes dégénérés ;
- $\sigma i = \sigma$ pour tout $\sigma \in C$;
- $\sigma r = \sigma$ si σ est un générateur induit par un simplexe dégénéré, sinon $\sigma r = 0_C$ pour tout $\sigma \in C^t$;
- $\partial = 0_C$;
- $\sigma j = 0_{C^{t+1}}$ si σ est un générateur induit par un simplexe dégénéré, sinon $\sigma j = \sigma$ pour tout $\sigma \in C^t$;
- $\sigma s = \sigma$ pour tout $\sigma \in C^{t+1}$.

L'équivalence homologique

$$\gamma : (C, \partial) \overset{\rho}{\Leftarrow} (C^B, \partial^B) \overset{\rho^S}{\Rightarrow} (C^S, \partial^S)$$

est calculée à partir de (C, ∂). L'équivalence homologique γ^{t+1} est calculée par application du cas 1 du théorème SECE.

Remarque. La preuve que S est une SECE est disponible en annexe E.

Exemple. La figure 4.12 illustre une SECE induite par la dégénérescence d'une arête e_0. Les morphismes i, j, r et s sont :

$i : C \to C^t$	$j : C^t \to C^{t+1}$	$r : C^t \to C$	$s : C^{t+1} \to C^t$
$e_0 i = e_0$	$e_0 j = 0$ $v_0 j = v_0$	$e_0 r = e_0$ $v_0 r = 0$	$v_0 s = v_0$

6. Pour rappel, l'association d'un complexe de chaînes à un ensemble simplicial est présentée dans le paragraphe 1.3.2-b.

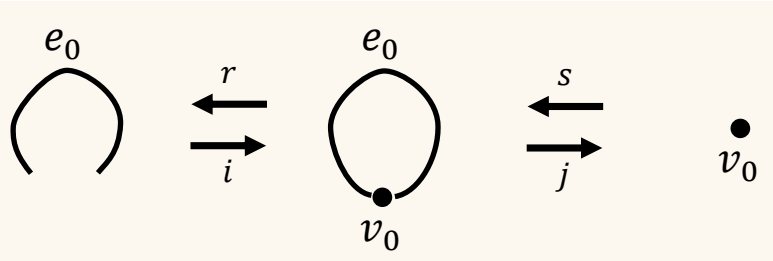


FIGURE 4.12 – Une SECE induite par la dégénérescence de e_0.

4.3.4-b Cadre commun

Pour commencer, revenons sur les structures topologiques sur lesquelles sont définies chacune des opérations :

— l'inclusion élémentaire et l'effondrement élémentaire sont définies sur les complexes simpliciaux. À notre connaissance, il n'existe pas d'extension des travaux sur la persistance de tours à d'autres structures topologiques. Rappelons que l'une des particularités des complexes simpliciaux est d'être entièrement définis à partir de leurs sommets ;

— l'identification est définie sur diverses structures topologiques, dont les ensembles simpliciaux.

Pour établir un lien entre ces opérations, il convient d'établir un lien entre les structures topologiques qu'elles impliquent. Pour cela, on utilise le fait que l'on sache construire un ensemble simplicial (K, d, s) à partir de tout complexe simplicial $\mathbb{K}$ en utilisant un algorithme comme celui présenté dans le paragraphe 1.2.1-c. En utilisant cette "conversion", on constate que :

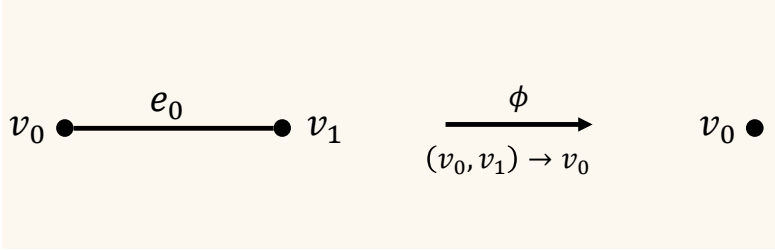
— l'inclusion élémentaire est simulée par l'ajout d'un simplexe et l'identification de tout ou partie de son bord avec des simplexes de (K, d, s) ;

— l'effondrement élémentaire est simulée par des identifications et des dégénérescences. En effet, en toute généralité, un effondrement élémentaire implique deux types de modifications sur $\mathbb{K}$,

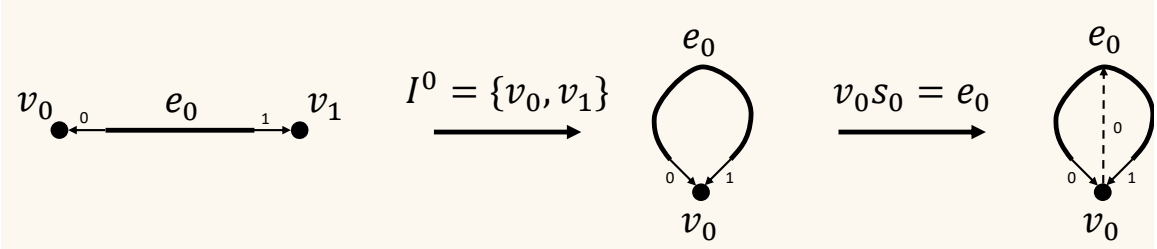
  · la fusion deux à deux de simplexes de même dimension (les simplexes miroirs en reprenant la terminologie utilisée dans le cas des tours monotones), qui peut être traitée sous la forme d'identifications ;

  · la disparition de certains simplexes (les simplexes fuyants), qui peut être traitée sous la forme de dégénérescences (cf. le paragraphe 1.2.1-c et plus particulièrement la figure 1.11).

  Cela est illustré sur la figure 4.13.



(a)



(b)

FIGURE 4.13 – Un effondrement élémentaire associant v_1 à v_0 et son équivalent sous la forme d'une identification et d'une dégénérescence dans l'ensemble simplicial correspondant. En particulier, notons que cet effondrement élémentaire n'a pas d'équivalent sur un ensemble semi-simplicial du fait de la disparition de l'arête $\{v_0, v_1\}$. (a) L'effondrement élémentaire. (b) L'identification et la dégénérescence.

4.3.4-c Exemple

La figure 4.14 illustre l'application successive de trois effondrements élémentaires $(v_1, v_4) \to v_1$, $(v_0, v_3) \to v_0$ et $(v_0, v_1) \to v_0$ sur un complexe simplicial $\mathbb{K}$. À l'aide de l'algorithme présenté dans le paragraphe 1.2.1-c, on associe un ensemble simplicial (K, d, s) à $\mathbb{K}$ et on étudie les implications des effondrements élémentaires sur ce dernier. Les opérations sur (K, d, s) induites par les effondrements élémentaires sont illustrées sur la figure 4.15.

Considérons $(v_1, v_4) \to v_1$. On observe que cet effondrement élémentaire n'implique que des simplexes miroirs : $\{v_1\}$ et $\{v_4\}$. Il peut être simulé par l'identification caractérisée par $I = \{\{v_1, v_4\}\}$.

Considérons $(v_0, v_3) \to v_0$. On observe que cet effondrement élémentaire n'implique que des simplexes miroirs : $\{v_0\}$ et $\{v_3\}$, $\{v_0, v_1\}$ et $\{v_3, v_1\}$. L'identification étant plus atomique que l'effondrement élémentaire, on peut simuler cet effondrement élémentaire d'au moins deux façons :

— en identifiant v_0 et v_3 et (v_0, v_1) et (v_3, v_1) en une passe ;

— en identifiant v_0 et v_3 puis (v_0, v_1) et (v_3, v_1), comme l'illustre la figure 4.15.

Enfin, considérons $(v_0, v_1) \to v_0$. On observe que cet effondrement élémentaire implique :

— des simplexes miroirs, en l'occurrence $\{v_0\}$ et $\{v_1\}$, $\{v_0, v_2\}$ et $\{v_1, v_2\}$, $\{v_1, v_5\}$ et $\{v_0, v_5\}$;

— des simplexes fuyants, en l'occurrence $\{v_0, v_1\}$, $\{v_0, v_1, v_2\}$ et $\{v_0, v_1, v_5\}$.

On peut simuler cet effondrement élémentaire en :

— identifiant les simplexes miroirs entre eux, c'est-à-dire en appliquant l'identification caractérisée par $I = \{\{v_0, v_1\}, \{(v_0, v_2), (v_1, v_2)\}, \{(v_0, v_5), (v_1, v_5)\}\}$;

— en dégénérant (v_0, v_1), (v_0, v_1, v_2) et (v_0, v_1, v_5) qui deviennent alors respectivement (v_0, v_0), (v_0, v_0, v_2) et (v_0, v_0, v_5), avec

  · $v_0 s_0 = (v_0, v_0)$

  · $(v_0, v_2) s_0 = (v_0, v_0, v_2)$;

  · $(v_0, v_5) s_0 = (v_0, v_0, v_5)$.

Remarque. Le procédé présenté ici se généralise à n'importe quel effondrement élémentaire. Dit autrement, pour simuler un effondrement élémentaire sur un ensemble simplicial, les simplexes miroirs sont d'abord identifiés entre eux, puis les simplexes fuyants sont dégénérés.

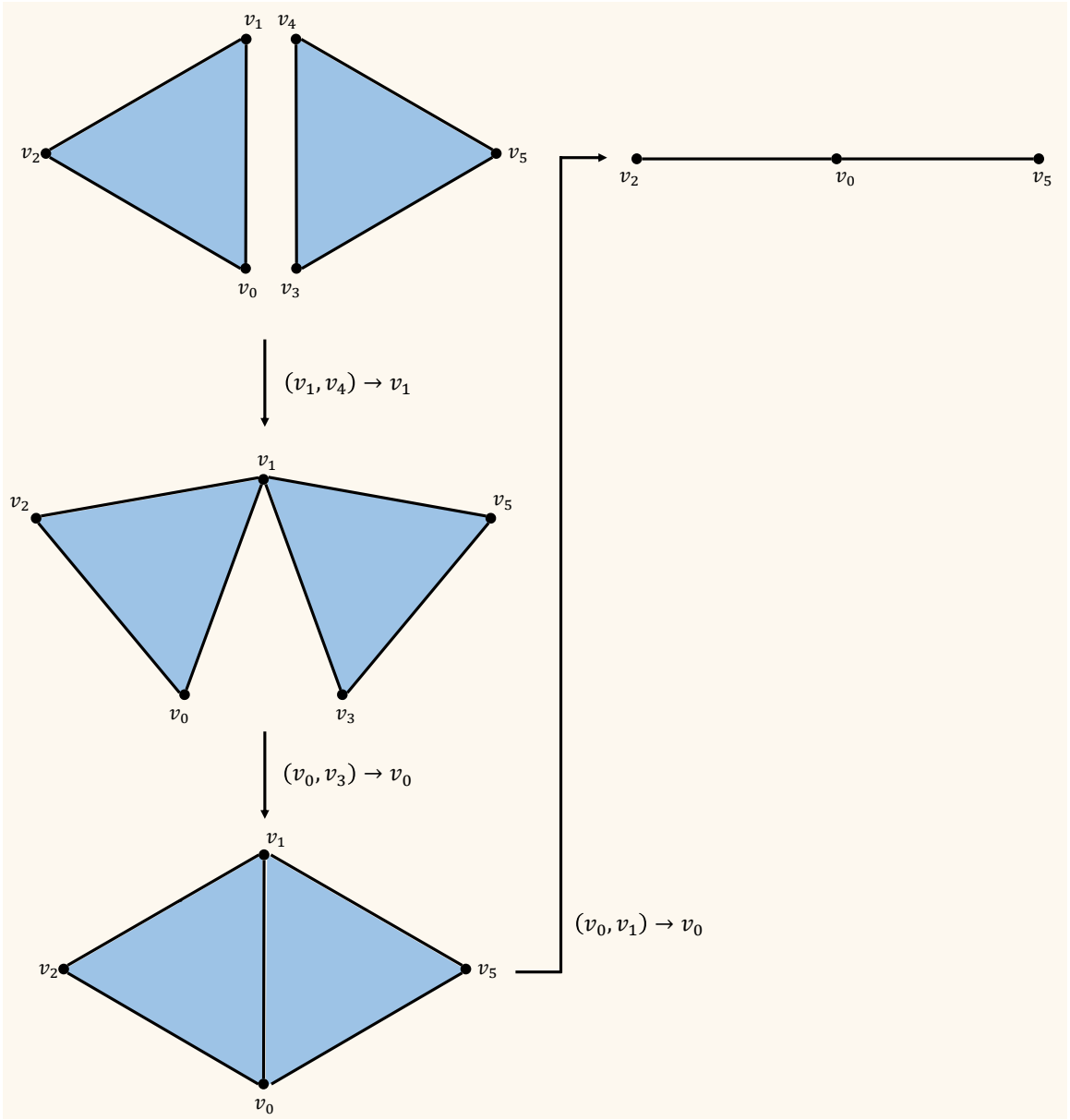


FIGURE 4.14 – Trois effondrements élémentaires $(v_1, v_4) \to v_1$, $(v_0, v_3) \to v_0$ et $(v_0, v_1) \to v_0$ appliqués successivement à un complexe simplicial $\mathbb{K}$.

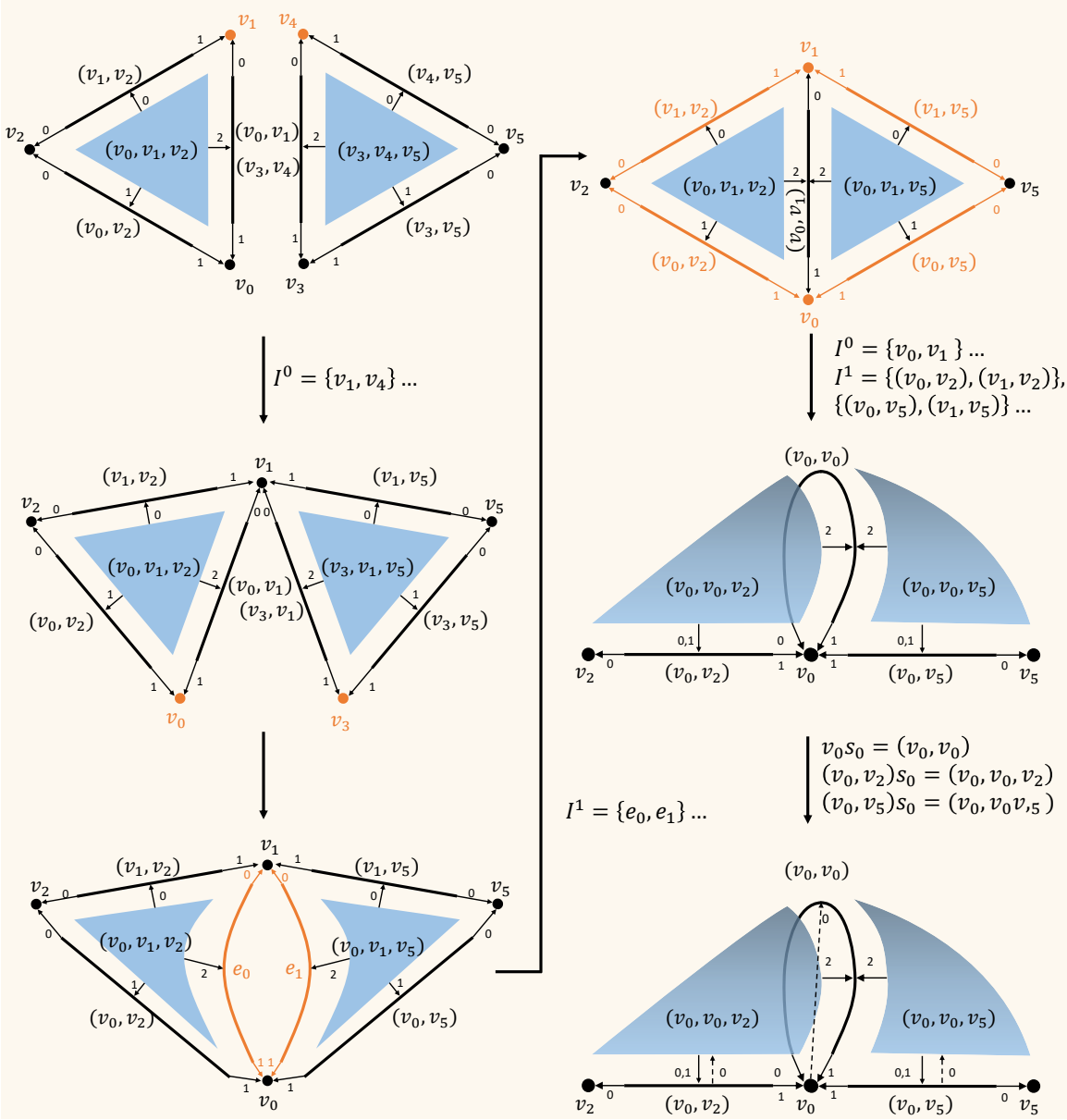


FIGURE 4.15 – Un équivalent des effondrements élémentaires illustrés sur la figure 4.14 sur l'ensemble simplicial (K, d, s) associé à $\mathbb{K}$.

4.3.5 Composition d'opérations et de leur inverse

Nous avons montré qu'il existe une correspondance entre les complexes simpliciaux et les opérations qui leur sont appliquées, à savoir l'effondrement et l'inclusion élémentaire, et les ensembles simpliciaux. Plus précisément, sur les ensembles simpliciaux :

— l'inclusion élémentaire est simulée par l'ajout de simplexe et l'identification ;

— l'effondrement élémentaire est simulé par l'identification et la dégénérescence.

Comme nous l'avons vu, l'identification et la dégénérescence induisent une SECE. Par conséquent, une équivalence homologique $\gamma^{t=0}$ peut être associée au complexe simplicial $\mathbb{K}_0$ d'une tour, et l'équivalence homologique γ^t associée à $\mathbb{K}_t$ peut être calculée par application du théorème SECE à partir des applications simpliciales $\phi_0 \dots \phi_t$ de cette même tour. En particulier :

— une tour monotone est simulée par une séquence de dégénérescences et d'identifications ;

— il en est de même pour une tour zigzag, sous réserve de traiter les effondrements élémentaires "zags" comme des dégénérescences et des identifications "en sens inverse" (et non comme des désidentifications).

Rappelons que le théorème SECE est applicable à la désidentification, ce qui veut dire qu'une tour zigzag peut être simulée en ne considérant qu'un seul sens de progression :

— un effondrement élémentaire "zig" (de $\mathbb{K}_t$ vers $\mathbb{K}_{t+1}$) peut être simulé par des dégénérescences et des identifications ;

— un effondrement élémentaire "zag" (de $\mathbb{K}_{t+1}$ vers $\mathbb{K}_t$) peut être simulé par des dégénérescences et des désidentifications.

Pour autant, dans cette sous-section, nous montrons que :

— nous montrons que deux SECE relevant du même cas [7] du théorème SECE se composent. Par exemple, deux SECE induites par deux identifications ou deux désidentifications ;

— à notre connaissance, deux SECE relevant de cas différents du théorème SECE ne se composent pas. Par exemple, une SECE induite par une identification et une SECE induite par une désidentification.

Cela veut dire qu'il est nécessaire, y compris pour la simulation avec le théorème SECE, de décomposer l'application successive d'un effondrement élémentaire "zig" et d'un effondrement élémentaire "zag" en deux opérations.

4.3.5-a Composition de deux SECE relevant du même cas du théorème.

Intuitivement, deux suites exactes courtes effectives qui relèvent du même cas du théorème SECE représentent deux opérations allant dans le même sens dans un processus de construction. En l'occurrence, il peut s'agir de deux identifications, deux désidentifications, deux dégénérescences voir d'une identification et d'une dégénérescence.

Définition 4.3.10. *Soient deux suites exactes courtes effectives*

$$S : (C, \partial) \underset{r}{\overset{i}{\rightleftarrows}} (C^t, \partial^t) \underset{s}{\overset{j}{\rightleftarrows}} (C^{t+1}, \partial^{t+1})$$

et

$$S' : (C', \partial') \underset{r'}{\overset{i'}{\rightleftarrows}} (C^{t+1}, \partial^{t+1}) \underset{s'}{\overset{j'}{\rightleftarrows}} (C^{t+2}, \partial^{t+2})$$

7. Pour rappel, les différents cas du théorème SECE sont présentés dans le paragraphe 1.4.3-a.

Leur composition

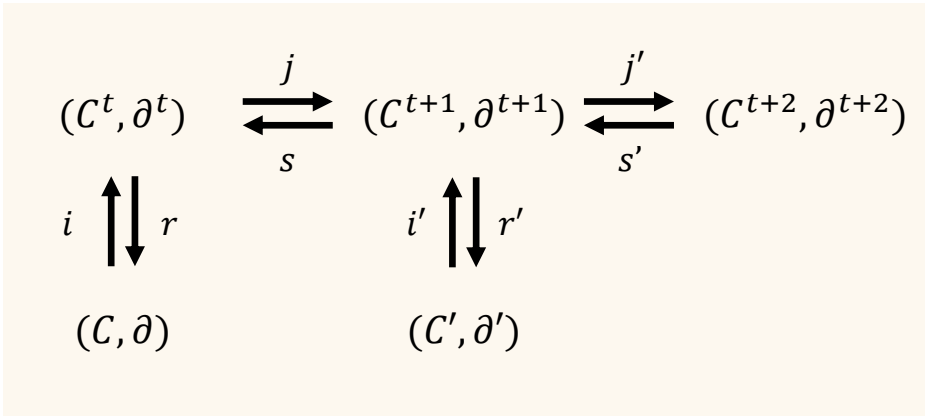
$$S'' : (C'', \partial'') \underset{r''}{\overset{i''}{\rightleftarrows}} (C^t, \partial^t) \underset{s''}{\overset{j''}{\rightleftarrows}} (C^{t+2}, \partial^{t+2})$$

est définie par :

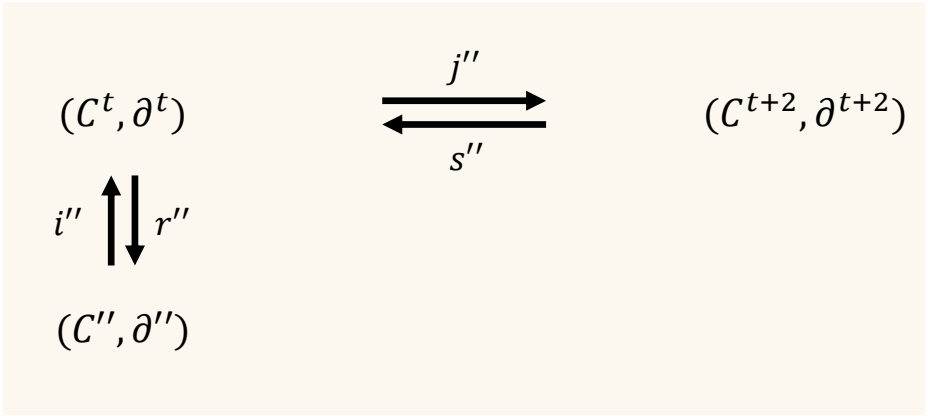
- $C'' = C \oplus C'$;
- $\partial'' = \partial + \partial' + \delta$ *avec* $\delta = i' s \partial^t r$;
- $i'' = i + i' s$;
- $r'' = r + j r'$;
- $j'' = j j'$;
- $s = s' s$;

Remarque. La preuve que S'' est une SECE est disponible en annexe F.

La figure 4.16 illustre les SECE d'une composition. Supposons de plus que l'on connaisse les équivalences homologiques γ et γ' associées respectivement à (C, ∂) et (C', ∂). Il est alors possible d'en déduire l'équivalence homologique γ'' associée à (C'', ∂'') par application des *lemmes de perturbation* présentés dans [16]. Cela permet d'optimiser le calcul de γ'' par rapport à un calcul direct à partir de (C'', ∂'').



(a)

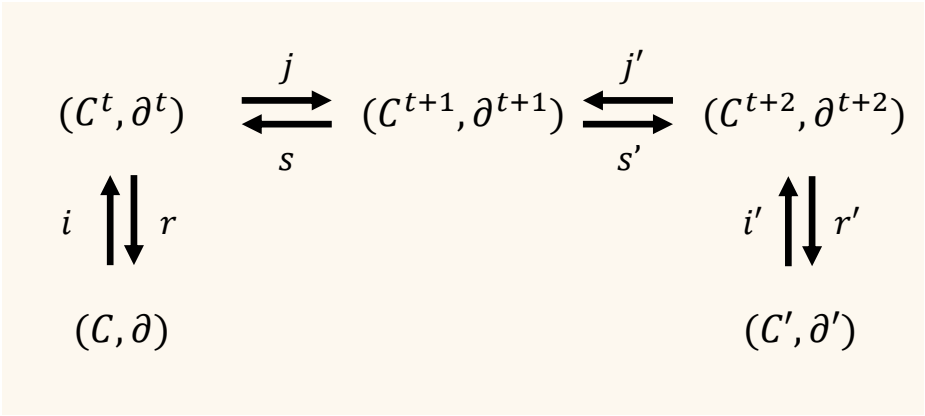


(b)

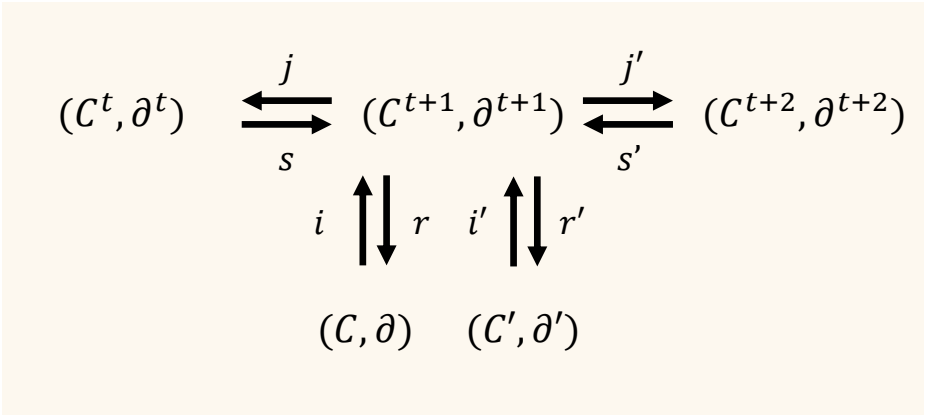
FIGURE 4.16 – Composition de deux suites exactes courtes effectives induites par un même cas du théorème SECE.

4.3.5-b Composition de deux SECE qui relèvent de cas différents du théorème.

Lorsque deux SECE relèvent de cas différents du théorème SECE, il n'est, à notre connaissance, pas possible de les composer. La figure 4.17 illustre les deux successions possibles de SECE relevant des cas 1 et 2 du théorème SECE. Dans les deux configurations, nous ne savons pas définir (C'', ∂'') de telle sorte à ce que S'' soit une SECE (c'est-à-dire qu'elle vérifie les propriétés mentionnées dans la définition 1.4.3-a).



(a)



(b)

FIGURE 4.17 – Les deux successions possibles de SECE relevant du cas 1 et du cas 2 du théorème. (a) Cas 1 puis cas 2. (b) Cas 2 puis cas 1.

4.4 Conclusion

Dans ce chapitre, nous étudions l'homologie persistante "de base" et "de tours" que nous mettons en correspondance avec les applications du théorème SECE à l'identification, la désidentification et la dégénérescence.

Homologie persistante de base. Nous montrons que la séquence de complexes de chaînes intermédiaires d'une équivalence homologique maintenue à l'aide du théorème SECE :

— induit une filtration dans le cas 1 du théorème (cf. paragraphe 1.4.3-a), utilisé dans son application à l'identification et la dégénérescence ;

— n'induit pas une filtration dans le cas 2, utilisé dans son application à la désidentification.

Pour autant, la désidentification étant l'inverse de l'identification, nous avons vu qu'il est possible d'associer une filtration "zigzag" à toute séquence d'applications du théorème SECE sous réserve de considérer deux sens de progression dans le processus de construction étudié. Plus précisément, cela revient à dire que la désidentification est traitée par application du 1 du théorème sur l'identification inverse qu'elle induit. Cette filtration peut servir de donnée d'entrée d'un algorithme d'homologie persistante ; l'intérêt d'un tel procédé reste à étudier.

Homologie persistante de tours. Nous établissons un lien entre le cadre de la persistance de tours et le cadre du théorème SECE, en montrant qu'il est possible de simuler un effondrement élémentaire appliqué à un complexe simplicial sous la forme d'identifications et de dégénérescence sur l'ensemble simplicial induit par ce dernier. En particulier, on constate qu'il existe des limitations à certains calculs dans les deux méthodes :

— les annotations sont utilisables tant que la tour étudiée est monotone, sinon une opération de cône simpliciale est utilisée ;

— on sait composer deux SECE relevant du même cas du théorème SECE mais on ne sait pas composer deux SECE relevant de cas différents du théorème SECE.

Perspectives d'évolution. L'objectif des travaux que nous présentons est d'établir un cadre commun à l'homologie persistante et au théorème SECE. Ce cadre peut servir de point de départ à une étude plus approfondie consistant à transférer des résultats d'un cadre à l'autre. Par exemple :

— peut-on éviter de maintenir la réduction ρ^S d'une équivalence homologique via le théorème SECE en utilisant la filtration induite par les complexes de chaînes intermédiaires ?

— comment peut-on utiliser l'application du théorème SECE à la désidentification pour simuler une tour zigzag ?

— les annotations peuvent-elles être transférées aux applications du théorème SECE ?

Conclusion et perspectives

Les chapitres 2, 3 et 4 disposent chacun de leur propre conclusion. Nous reprenons les points importants de chacune d'elles.

Conclusion

Pour suivre l'évolution de l'homologie d'un objet géométrique dans un processus de construction, nous utilisons le théorème des suites exactes courtes effectives (théorème SECE). Plus précisément, on suppose qu'une équivalence homologique $\gamma^{t=0}$ est associée à l'objet géométrique à l'étape initiale du processus, et on la maintient à l'aide du théorème SECE. Son petit complexe de chaînes peut être utilisé, à chaque étape, pour calculer l'homologie de l'objet géométrique. Nous supposons que le passage de l'étape t à l'étape $t+1$ du processus se fait par identification ou par désidentification (éventuellement, par dégénérescence).

Nous analysons l'application du théorème SECE à ces opérations et montrons que γ^{t+1} peut être calculée par modification de γ^t. En particulier, plusieurs cas se distinguent :

— les cas "limités", où seule une partie des matrices de γ^t est maintenue, par exemple, lorsqu'on ne souhaite pas calculer les générateurs d'homologie et que le processus de construction n'est composé que d'identifications ;

— le cas général, où toutes les matrices de γ^t doivent être maintenues, par exemple, lorsqu'on souhaite calculer les générateurs d'homologie et que le processus de construction est composé d'au moins une identification.

Pour les cas limités, on observe que :

— le calcul de γ^{t+1} par modification de γ^t a une complexité en temps qui dépend du nombre de cellules impactées par l'opération, et de leur étoile ;

— le coût en espace de γ^{t+1} est égal au coût de γ^t auquel s'ajoute une variation fonction du nombre de cellules impactées par l'opération. Plus généralement, le coût en espace de γ^t est égal au coût de $\gamma^{t=0}$ auquel s'ajoute la somme des variations des étapes qui précèdent t ;

Pour le cas général, on observe le même type de résultat sauf pour le calcul de $h^{S,t+1}$, qui a une complexité en temps qui, dans le pire des cas, dépend du nombre total de cellules de la structure topologique.

Nous mettons en évidence des prérequis que doit vérifier une implémentation du théorème SECE pour obtenir ces résultats en pratique. Les morphismes sont représentés sous la forme de matrices. Les prérequis se regroupent en plusieurs catégories :

— ceux concernant la représentation matricielle. Les valeurs nulles d'une matrice sont représentées implicitement de même que l'identité dans certains cas ;

— ceux concernant les calculs. Cela comprend des contraintes sur la construction des matrices, sur le produit matriciel, sur la fusion de matrices ou encore sur la modification de matrices.

Après avoir étudié plusieurs bibliothèques logicielles dédiées à la représentation matricielle, nous constatons que la plupart d'entre elles implémentent des formats "standards". Ces derniers ne permettent pas de répondre à l'ensemble des prérequis d'implémentation que nous avons relevés.

Nous avons donc conçu et développé notre propre représentation matricielle, qui fonctionne de pair avec des répertoires de cellules. Un répertoire de cellules représente une base des chaînes d'un complexe de chaînes de γ^t depuis l'étape initiale du processus de construction. En utilisant ces structures de données, nous avons développé une bibliothèque logicielle d'homologie effective que nous utilisons pour étudier expérimentalement le théorème SECE. Les résultats ainsi obtenus valident l'analyse menée en amont.

Nous nous intéressons ensuite au suivi de l'homologie d'un objet géométrique lorsque celui-ci est éclaté en plusieurs morceaux distribués sur plusieurs unités de calcul. Dans ce cadre, nous proposons un algorithme permettant de répartir l'identification "appliquée" à l'objet géométrique sur sa distribution. Le développement du logiciel implémentant cet algorithme reste à terminer. Pour autant, nous présentons les travaux que nous avons réalisés à ce sujet, basés sur le standard MPI.

Pour finir, nous mettons en corrélation le théorème SECE et l'homologie persistante. Cette dernière repose sur la notion de filtration. Nous montrons que le théorème SECE induit une filtration lorsqu'il est appliqué à l'identification et que ce n'est pas le cas lorsqu'il est appliqué à la désidentification. Nous étudions ensuite une extension de l'homologie persistante, l'homologie persistante de tours, et montrons qu'il existe un cadre commun entre les opérations élémentaires qu'elle traite et celles que nous traitons. En particulier, on observe que les critères de discriminations des cas du théorème SECE et de l'homologie persistante de tours se rejoignent sur un point : le "sens" de progression dans le processus de construction. Plus précisément, on observe que, dans les deux cas, les calculs peuvent être optimisés tant que la progression est monotone.

Perspectives d'évolutions

Optimiser l'espace. Nous représentons implicitement une partie des données d'une équivalence homologique

$$\gamma^t : (C^t, \partial^t) \overset{\rho^t}{\Longleftarrow} (C^{B,t}, \partial^{B,t}) \overset{\rho^{S,t}}{\Longrightarrow} (C^{S,t}, \partial^{S,t})$$

maintenue par application du théorème SECE : l'identité et le morphisme nul. En particulier, nous tirons parti du fait que $(C^{t=0}, \partial^{t=0}) = (C^{B,t=0}, \partial^{B,t=0})$ pour représenter f^t et g^t comme des variations de l'identité car, à $t = 0$, ρ^t est une réduction identité et donc $f^t = g^t = id$. Cette représentation implicite de l'identité pourrait être étendue aux complexes de chaînes (C^t, ∂^t) et $(C^{B,t}, \partial^{B,t})$:

— d'abord, $(C^{B,t}, \partial^{B,t})$ pourrait être considéré comme une variation de (C^t, ∂^t) ;

— ensuite, $(C^{B,t}, \partial^{B,t})$ et (C^t, ∂^t) pourraient être considérés comme une variation de $(C^{t=0}, \partial^{t=0})$.

Nous pensons qu'un tel procédé permettrait d'optimiser l'utilisation de l'espace pour la représentation de γ^t.

Analyser la notion de réduction. La réduction est une notion clef du théorème SECE puisqu'elle compose les équivalences homologiques manipulées. Plusieurs réductions peuvent donner "un même" complexe de chaînes réduit. Pour autant, les morphismes qui les composent peuvent être très différents d'une telle réduction à l'autre, et ceci est particulièrement vrai pour le morphisme h. Une façon de faire évoluer les travaux que nous présentons consiste à déterminer si certaines réductions sont meilleures que d'autres pour les calculs qu'implique le théorème SECE (cf. annexe C).

Extension de l'algorithme de répartition. Dans le cas d'un objet distribué, pour couvrir l'ensemble des opérations traitées par le théorème SECE, il manque à ce jour un algorithme de répartition de désidentification. Le concevoir est une évolution qui permettrait d'étudier la répartition d'opérations dans un processus de construction composé uniquement d'identifications, uniquement de désidentifications, ou des deux opérations.

Alimenter l'étude expérimentale. L'étude expérimentale que nous proposons est basée sur l'application du théorème SECE à l'identification pour un objet géométrique non distribué. Nous souhaitons l'alimenter et proposer l'étude expérimentale de l'application du théorème SECE à :

— la désidentification pour un objet non distribué. L'analyse montre que les prérequis sont sensiblement les mêmes que pour l'identification. L'objectif est d'appuyer ce résultat expérimentalement ;

— l'identification (et, si possible, d'une désidentification) pour un objet distribué. L'objectif est de valider l'algorithme de répartition.

Approfondir l'étude des tours et du théorème SECE. Nous montrons qu'il existe des liens entre l'homologie persistante et le théorème SECE :

— une séquence d'opérations relevant du cas 1 du théorème SECE induit une filtration entre les complexes de chaînes intermédiaires de l'équivalence homologique maintenue ;

— nous définissons un cadre commun aux tours et au théorème SECE, qui repose sur le fait que l'on sache construire un ensemble simplicial à partir d'un complexe simplicial, et que les applications simpliciales d'une tour peuvent être simulées sous la forme d'identifications et de dégénérescences.

Ces liens peuvent servir de point de départ à une étude plus approfondie de l'homologie persistante et du théorème SECE, dont l'objet serait de transférer les résultats d'un cadre à l'autre (éviter le maintien de $\rho^{S,t}$, utiliser des coefficients dans $\mathbb{Z}$ etc. . .).

Bibliographie

[1] Julio Rubio and Francis Sergeraert. Constructive homological algebra and applications. Technical report, Universidad de la Rioja, Université Grenoble Alpes, 2006.

[2] Tamal K Dey, Fengtao Fan, and Yusu Wang. Computing topological persistence for simplicial maps. In *Proceedings of the thirtieth annual symposium on Computational geometry*, pages 345–354, 2014.

[3] Robin J.WILSON. *Introduction to Graph Theory*. Longman Group Ltd, 4 edition, 1996.

[4] Guillaume Damiand and Rocio Gonzalez-Diaz. Parallel homology computation of meshes. In *International Workshop on Computational Topology in Image Context*, pages 53–64. Springer, 2016.

[5] Gadea Mata Martínez. *Processing biomedical images for the study of treatments related to neurodegenerative diseases*. PhD thesis, Universidad de La Rioja, 2017.

[6] Daniel Harlow and Hirosi Ooguri. Symmetries in quantum field theory and quantum gravity. *Communications in Mathematical Physics*, 383(3) :1669–1804, 2021.

[7] Peter Bubenik et al. Statistical topological data analysis using persistence landscapes. *J. Mach. Learn. Res.*, 16(1) :77–102, 2015.

[8] Julien Tierny, Guillaume Favelier, Joshua A. Levine, Charles Gueunet, and Michael Michaux. The topology toolkit. *IEEE Transactions on Visualization and Computer Graphics*, 24(1) :832–842, 2018.

[9] Afra Zomorodian. Topological data analysis. *Advances in applied and computational topology*, 70 :1–39, 2012.

[10] Darian M. Onchis, Codruta Istin, and Pedro Real. Refined deep learning for digital objects recognition via betti invariants. In Mario Vento and Gennaro Percannella, editors, *Computer Analysis of Images and Patterns*, pages 613–621, Cham, 2019. Springer International Publishing.

[11] Peter Bürgisser, Felipe Cucker, and Josué Tonelli-Cueto. Computing the homology of semialgebraic sets. ii : General formulas. *Foundations of Computational Mathematics*, pages 1–38, 2021.

[12] Andrea Guidolin, Jose Divasón, Ana Romero, and Francesco Vaccarino. Computing invariants for multipersistence via spectral systems and effective homology. *Journal of Symbolic Computation*, 104 :724–753, 2021.

[13] Li Chen. Algorithms for computing topological invariants in 2d and 3d digital spaces. *CoRR*, abs/1309.4109, 2013.

[14] Michelle Feng and Mason A Porter. Persistent homology of geospatial data : A case study with voting. *SIAM Review*, 63(1) :67–99, 2021.

[15] Tamal K. Dey and Tao Hou. Computing zigzag persistence on graphs in near-linear time. *CoRR*, abs/2103.07353, 2021.

[16] Sylvie Alayrangues, Laurent Fuchs, Pascal Lienhardt, and Samuel Peltier. Homology computation during an incremental construction of simplicial or cellular structures. Working paper or preprint, April 2016.

[17] Wassim Rharbaoui, Sylvie Alayrangues, Pascal Lienhardt, and Samuel Peltier. Local computation of homology variations over a construction process. *Computer Aided Geometric Design*, 81 :101907, 2020.

[18] Serge Lang. *Algebra*, volume 1 of *Graduate Texts in Mathematics*. Springer, 2002.

[19] Michael Kerber and Hannah Schreiber. Barcodes of towers and a streaming algorithm for persistent homology. *Discrete & computational geometry*, 61(4) :852–879, 2019.

[20] Samuel Peltier. *Calcul de groupes d'homologie sur des structures simpliciales, simploïdales et cellulaires*. PhD thesis, Faculté des sciences fondamentales et appliquées, 2006.

[21] J Peter May. *Simplicial objects in algebraic topology*, volume 11. University of Chicago Press, 1992.

[22] Veronique Lang and Pascal Lienhardt. Simplicial sets and triangular patches. In *Proceedings of CG International'96*, pages 154–163. IEEE, 1996.

[23] Samuel Peltier and Pascal Lienhardt. Simploidal Sets : a data structure for handling simploidal Bezier spaces. *Computer Aided Geometric Design*, 62 :44 – 62, May 2018.

[24] Sylvie Alayrangues, Pascal Lienhardt, and Samuel Peltier. Conversion between chains of maps and chains of surfaces ; application to the computation of incidence graphs homology. Research report, Université de Poitiers, 2015.

[25] Pascal Lienhardt. N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *International Journal of Computational Geometry and Applications*, 4(03) :275–324, 1994.

[26] Guillaume Damiand and Pascal Lienhardt. *Combinatorial maps : efficient data structures for computer graphics and image processing*. CRC Press, 2014.

[27] Akhmet Aksanovich Khusainov. Homology groups of semicubical sets. *Siberian Mathematical Journal*, 49(1) :180–190, 2008.

[28] Sylvie Alayrangues, Samuel Peltier, Guillaume Damiand, and Pascal Lienhardt. Border operator for generalized maps. In Srečko Brlek, Christophe Reutenauer, and Xavier Provençal, editors, *Discrete Geometry for Computer Imagery*, pages 300–312, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[29] David Cohen-Steiner, André Lieutier, and Julien Vuillamy. Lexicographic optimal homologous chains and applications to point cloud triangulations. In *36th International Symposium on Computational Geometry (SoCG 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[30] Ippei Obayashi. Volume-optimal cycle : Tightest representative cycle of a generator in persistent homology. *SIAM Journal on Applied Algebra and Geometry*, 2(4) :508–534, 2018.

[31] Pengxiang Wu, Chao Chen, Yusu Wang, Shaoting Zhang, Changhe Yuan, Zhen Qian, Dimitris Metaxas, and Leon Axel. Optimal topological cycles and their application in cardiac trabeculae restoration. In *International Conference on Information Processing in Medical Imaging*, pages 80–92. Springer, 2017.

[32] Tamal K Dey, Anil N Hirani, and Bala Krishnamoorthy. Optimal homologous cycles, total unimodularity, and linear programming. *SIAM Journal on Computing*, 40(4) :1026–1044, 2011.

[33] Paul Moritz Cohn. *Algebra*, volume 1. J. Wiley and sons, 2 edition, 1982.

[34] Munkres James R. *Elements of algebraic topology*. Addison-Wesley, 1984.

[35] Henry John Stephen Smith. On systems of linear indeterminate equations and congruences. *Philosophical Transactions of the Royal Society of London*, 151 :293–326, 1861.

[36] Jean-Guillaume Dumas, B David Saunders, and Gilles Villard. On efficient sparse integer matrix smith normal form computations. *Journal of Symbolic Computation*, 32(1-2) :71–99, 2001.

[37] David Saunders and Zhendong Wan. Smith normal form of dense integer matrices fast algorithms into practice. In *Proceedings of the 2004 international symposium on Symbolic and algebraic computation*, pages 274–281, 2004.

[38] Stavros Birmpilis, George Labahn, and Arne Storjohann. A las vegas algorithm for computing the smith form of a nonsingular integer matrix. In *Proceedings of the 45th International Symposium on Symbolic and Algebraic Computation*, pages 38–45, 2020.

[39] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge university press, 1995.

[40] Wayne Eberly, Mark Giesbrecht, and Gilles Villard. On computing the determinant and smith form of an integer matrix. *Computer Science*, 2000.

[41] 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.

[42] Arne Storjohann. Algorithms for matrix canonical forms. *A dissertation for the degree of Doctor of Technical Sciences*, 2013.

[43] Jose Divasón. A verified algorithm for computing the smith normal form of a matrix. *Arch. Formal Proofs*, 2020, 2020.

[44] N Anurag Murty, Vijay Natarajan, and Sathish Vadhiyar. Efficient homology computations on multicore and manycore systems. In *20th Annual International Conference on High Performance Computing*, pages 333–342. IEEE, 2013.

[45] Shaun Harker, Konstantin Mischaikow, Marian Mrozek, and Vidit Nanda. Discrete morse theoretic algorithms for computing homology of complexes and maps. *Foundations of Computational Mathematics*, 14(1) :151–184, 2014.

[46] Max K Agoston. *Algebraic topology : a first course*, volume 32. M. Dekker, 1976.

[47] Richard P Stanley. Smith normal form in combinatorics. *Journal of Combinatorial Theory, Series A*, 144 :476–495, 2016.

[48] RB Bapat and Sivaramakrishnan Sivasubramanian. Smith normal form of a distance matrix inspired by the four-point condition. *Linear Algebra and its Applications*, 2020.

[49] CN Umadevi and NP Gopalan. Outsourcing computations through smith normal form with access control. In *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, pages 350–353. IEEE, 2019.

[50] Luis Elcoro, Benjamin J Wieder, Zhida Song, Yuanfeng Xu, Barry Bradlyn, and B Andrei Bernevig. Magnetic topological quantum chemistry. *arXiv preprint arXiv :2010.00598*, 2020.

[51] Herbert Edelsbrunner and John Harer. *Computational topology : an introduction*. American Mathematical Soc., 2010.

[52] Gunnar Carlsson and Vin De Silva. Zigzag persistence. *Foundations of computational mathematics*, 10(4) :367–405, 2010.

[53] Nikola Milosavljević, Dmitriy Morozov, and Primoz Skraba. Zigzag persistent homology in matrix multiplication time. In *Proceedings of the twenty-seventh annual symposium on Computational geometry*, pages 216–225, 2011.

[54] Clément Maria and Hannah Schreiber. Discrete morse theory for computing zigzag persistence. In *Workshop on Algorithms and Data Structures*, pages 538–552. Springer, 2019.

[55] Gunnar Carlsson and Afra Zomorodian. The theory of multidimensional persistence. *Discrete & Computational Geometry*, 42(1) :71–93, 2009.

[56] Madjid Allili, Tomasz Kaczynski, and Claudia Landi. Reducing complexes in multidimensional persistent homology theory. *Journal of Symbolic Computation*, 78 :61–75, 2017.

[57] Sara Scaramuccia, Federico Iuricich, Leila De Floriani, and Claudia Landi. Computing multiparameter persistent homology through a discrete morse-based approach. *Computational Geometry*, 89 :101623, 2020.

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

[59] Jean-Daniel Boissonnat and Clément Maria. Computing Persistent Homology with Various Coefficient Fields in a Single Pass. *Journal of Applied and Computational Topology*, 3(1-2) :16, September 2019.

[60] Konstantin Mischaikow and Vidit Nanda. Morse theory for filtrations and efficient computation of persistent homology. *Discrete & Computational Geometry*, 50(2) :330–353, 2013.

[61] Paweł Dłotko and Hubert Wagner. Computing homology and persistent homology using iterated morse decomposition. *arXiv preprint arXiv :1210.1429*, 2012.

[62] Paweł Dłotko, Hubert Wagner, et al. Simplification of complexes for persistent homology computations. *Homology, Homotopy and Applications*, 16(1) :49–63, 2014.

[63] Henry King, Kevin Knudson, and Neža Mramor. Generating discrete morse functions from point data. *Experimental Mathematics*, 14(4) :435–444, 2005.

[64] Donald R Sheehy. Linear-size approximations to the vietoris–rips filtration. *Discrete & Computational Geometry*, 49(4) :778–796, 2013.

[65] Magnus Bakke Botnan and Gard Spreemann. Approximating persistent homology in euclidean space through collapses. *Applicable Algebra in Engineering, Communication and Computing*, 26(1-2) :73–101, 2015.

[66] Tamal K Dey, Dayu Shi, and Yusu Wang. Simba : An efficient tool for approximating rips-filtration persistence via simplicial batch collapse. *Journal of Experimental Algorithmics (JEA)*, 24(1) :1–16, 2019.

[67] Michael Kerber and R Sharathkumar. Approximate čech complex in low and high dimensions. In *International Symposium on Algorithms and Computation*, pages 666–676. Springer, 2013.

[68] Herbert Edelsbrunner, Ziga Virk, and Hubert Wagner. Topological data analysis in information space. *arXiv preprint arXiv :1903.08510*, 2019.

[69] Gunnar Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2) :255–308, 2009.

[70] Andrew Aukerman, Mathieu Carrière, Chao Chen, Kevin Gardner, Raúl Rabadán, and Rami Vanguri. Persistent homology based characterization of the breast cancer immune microenvironment : A feasibility study. In *36th International Symposium on Computational Geometry (SoCG 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[71] Paul Bendich, James S Marron, Ezra Miller, Alex Pieloch, and Sean Skwerer. Persistent homology analysis of brain artery trees. *The annals of applied statistics*, 10(1) :198, 2016.

[72] Kelin Xia and Guo-Wei Wei. Multidimensional persistence in biomolecular data. *Journal of computational chemistry*, 36(20) :1502–1520, 2015.

[73] Kelin Xia and Guo-Wei Wei. Persistent homology analysis of protein structure, flexibility, and folding. *International journal for numerical methods in biomedical engineering*, 30(8) :814–844, 2014.

[74] Bryn Keller, Michael Lesnick, and Theodore L. Willke. Persistent Homology for Virtual Screening. *ChemRxiv*, 10 2018.

[75] Herbert Edelsbrunner and Ernst P Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics (TOG)*, 13(1) :43–72, 1994.

[76] Stefan Dantchev and Ioannis Ivrissimtzis. Efficient construction of the čech complex. *Computers & Graphics*, 36(6) :708–713, 2012.

[77] Afra Zomorodian. Fast construction of the vietoris-rips complex. *Computers & Graphics*, 34(3) :263–271, 2010.

[78] Dobrina Boltcheva, Sara Merino Aceitunos, Jean-Claude Léon, and Franck Hétroy. Constructive mayer-vietoris algorithm : Computing the homology of unions of simplicial complexes. Research Report RR-7471, INRIA, December 2010.

[79] Dobrina Boltcheva, David Canino, Sara Merino Aceituno, Jean-Claude Léon, Leila De Floriani, and Franck Hétroy. An iterative algorithm for homology computation on simplicial shapes. *Computer-Aided Design*, 43(11) :1457–1467, 2011.

[80] Aldo Gonzalez Lorenzo. *Computational homology applied to discrete objects*. PhD thesis, Université d'Aix-Marseille, Universidad de Sevilla, 2016. Thèse de doctorat dirigée par Mari, Jean-LucBac-Bruasse, Alexandra et Real Jurado, Pedro Informatique Aix-Marseille, Universidad de Sevilla (Espagne) 2016.

[81] Oleksiy Busaryev, Sergio Cabello, Chao Chen, Tamal K Dey, and Yusu Wang. Annotating simplices with a homology basis and its applications. In *Scandinavian workshop on algorithm theory*, pages 189–200. Springer, 2012.

[82] Chao Chen and Michael Kerber. An output-sensitive algorithm for persistent homology. *Computational Geometry*, 46(4) :435–447, 2013.

[83] Jean-Daniel Boissonnat and Siddharth Pritam. Computing persistent homology of flag complexes via strong collapses. Technical report, Inria, 2019.

[84] Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 522–539. SIAM, 2021.

[85] Alberto Parravicini, Francesco Sgherzi, and Marco D Santambrogio. A reduced-precision streaming spmv architecture for personalized pagerank on fpga. In *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 378–383. IEEE, 2021.

[86] Intel. Intel® oneapi math kernel library - developer reference, 2021.2. Documentation IntelMKL.

[87] Standard C++ Foundation. Programming languages — C++. Standard ISO 14882 :2020(E), International Organization for Standardization, 2020.

[88] Guillaume Damiand, Aldo Gonzalez-Lorenzo, Florence Zara, and Florent Dupont. Distributed combinatorial maps for parallel mesh processing. *Algorithms*, 11(7) :105, 2018.

[89] MPI Forum. MPI : A Message-Passing Interface Standard. Technical report, MPI Forum, 2021.

Annexes

A Réduction élémentaire

La définition ci-dessous est une adaptation de celle de l'annexe 6.1 de [16].

Définition A.1. *Soient un complexe de chaînes (C, ∂) et deux chaînes $\sigma \in C^p$ et $\tau \in C^{p+1}$ tels que σ apparaît dans le bord de τ avec un coefficient $\epsilon = \pm 1$. La réduction élémentaire caractérisée par σ et τ est une réduction $\rho : (C, \partial) \Rrightarrow (C', \partial')$ telle que :*

- *$h : C \to C$ est un morphisme nul, sauf pour $\sigma h = \epsilon\tau$;*
- *$f : C \to C'$ est l'identité, sauf pour*
  - *$\sigma f = (-\epsilon\tau\partial + \sigma)\phi^{-1}$*
  - *$\tau f = 0_{C'}$*
- *$g : C' \to C$ est, pour tout $\lambda \in C'$,*
  - *$\lambda g = \lambda\phi - \lambda\phi\partial h$*
- *∂' est, pour tout $\lambda \in C''$,*
  - *$\lambda\partial' = \lambda\phi^{-1}\partial f$*

où ϕ est l'inclusion de $B_{C'}$ dans B_C.

Exemple. La figure 18 illustre une réduction élémentaire $\rho : (C, \partial) \Rrightarrow (C', \partial')$ telle que :

- $h : C \to C$ est un morphisme nul, sauf pour $(a_0, v_2)h = (a_0, a_1)$;
- $f : C \to C'$ est l'identité, sauf pour
  - $(a_0, v_2)f = (-(a_0, a_1)\partial + (a_0, v_2))\phi^{-1} = -(v_0, a_1) + (a_0, v_1) + (v_1, a_1)$
  - $(a_0, a_1)f = 0_{C'}$
- $g : C' \to C$ est l'inclusion de C' dans C. Il s'agit là d'un cas particulier qui découle du fait que (a_0, a_1) soit l'unique générateur de C' ayant (a_0, v_2) dans son bord.

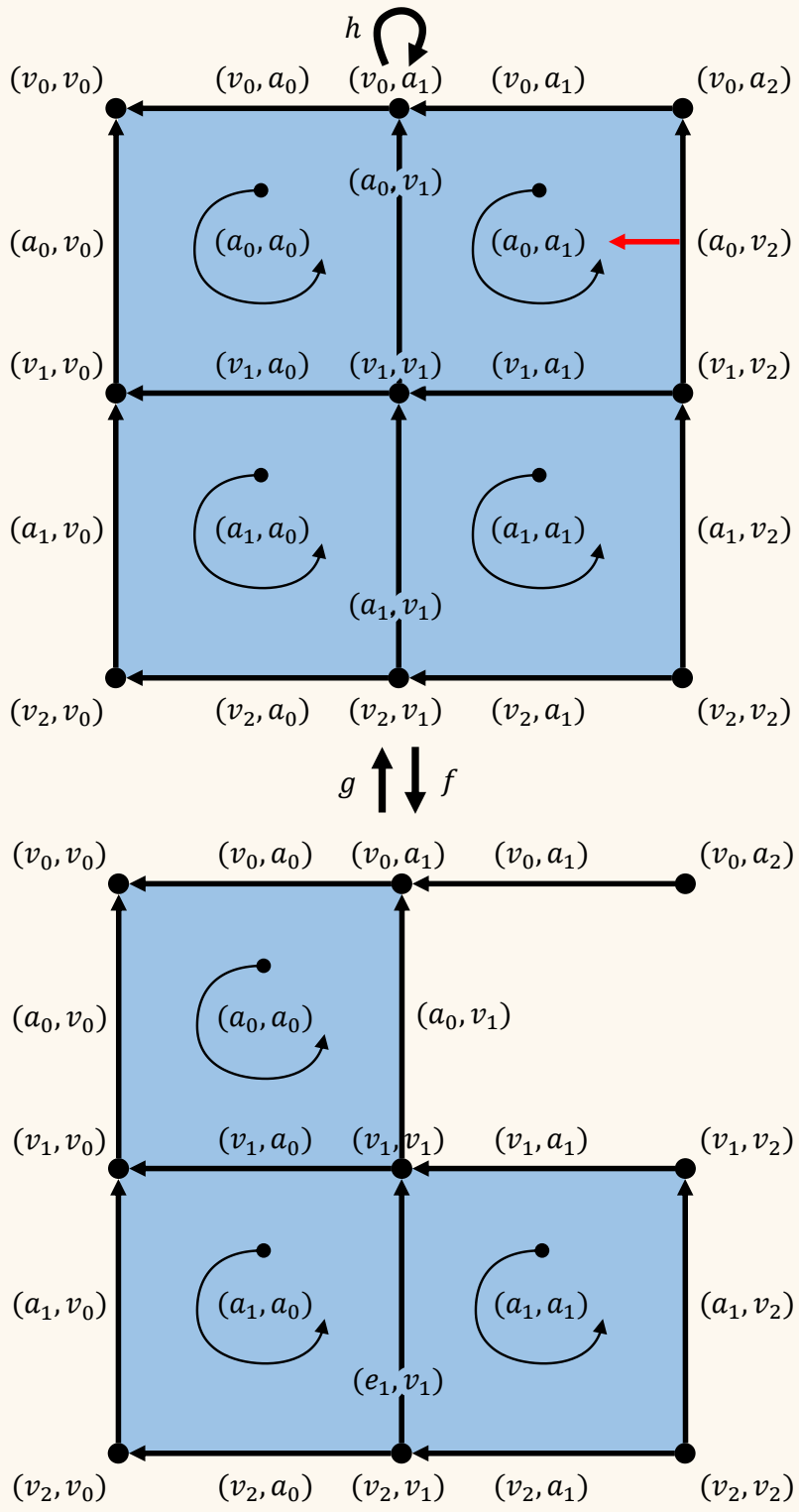


FIGURE 18 – Une réduction élémentaire dans une $(2, 2)$-grille, telle que $(a_0, v_2)h = (a_0, a_1)$.

B Composition de réduction

La définition ci-dessous est une adaptation de celle de [16] page 6.

Définition B.1. *Soient une réduction* $\rho : (C, \partial) \Rrightarrow (C', \partial')$ *et* $\rho' : (C', \partial') \Rrightarrow (C'', \partial'')$. *Leur composition est une réduction* $\rho'' : (C, \partial) \Rrightarrow (C'', \partial'')$ *définie par :*

— $h'' = h + fh'g$

— $f'' = ff'$

— $g'' = g'g$

Exemple. La figure 19 illustre une composition de réductions. En particulier :

— $(a_0, v_2)h'' = (a_0, a_1) + (a_0, a_0)$ et $(a_0, v_1)h'' = (a_0, a_0)$;

— $(a_0, v_2)f'' = -(v_0, a_1) - (v_0, a_0) + (a_0, v_0) + (v_1, a_0) + (v_1, a_1)$ et $(a_0, v_1)f'' = -(v_0, a_0) + (a_0, v_0) + (v_1, a_0)$;

— $g : C'' \to C$ est l'inclusion de C'' dans C. Il s'agit d'un cas particulier qui découle du fait que (a_0, a_1) soit l'unique générateur de C ayant (a_0, v_2) dans son bord, et que (a_0, a_0) soit l'unique générateur de C' ayant (a_0, v_1) dans son bord.

C À propos de la réduction

Dans cette section, on s'intéresse aux "chemins" de réduction créés par une composition, et l'idée de "réduction optimale" qui en découle.

Chemin de réduction. Soient une réduction élémentaire $\rho : (C, \partial) \Rightarrow (C', \partial')$ caractérisée par le couple de cellules (τ, σ), et une réduction élémentaire $\rho' : (C', \partial') \Rightarrow (C'', \partial'')$ caractérisée par le couple de cellules (τ', σ'), où τ et τ' sont des p-cellules et σ et σ' sont des $(p-1)$-cellules. Considérons leur composition $\rho'' : (C, \partial) \Rightarrow (C'', \partial'')$. Observons que :

— pour n'importe quelle cellule de (C, ∂), f'' est l'identité sauf pour $\tau f'' = \tau' f'' = 0_{C''}$ et pour σ et σ'. Intéressons-nous à ces dernières :

· dans ρ, on a $\sigma f = (-\epsilon\tau\partial + \sigma)$, où ϵ désigne le coefficient avec lequel σ apparaît dans le bord de τ dans (C, ∂) ;

· dans ρ', on a $\sigma' f' = (-\epsilon\tau'\partial' + \sigma')$, où ϵ désigne le coefficient avec lequel σ' apparaît dans le bord de τ' dans (C', ∂') ;

· f'' est calculée via la formule $f'' = ff'$. Or, f' est l'identité pour toute cellule sauf $\tau' f' = 0_{C''}$ et $\sigma' f' = (-\epsilon\tau'\partial' + \sigma')$. Les cellules pour lesquelles $f'' \neq f'$ sont donc toutes les cellules qui ont τ' ou σ' dans leur image par f ;

· f est l'identité pour toutes les cellules sauf $\tau f = 0_{C'}$ et $\sigma f = (-\epsilon\tau\partial + \sigma)$. La seule cellule qui a τ' dans son image par f est τ', or $\tau' f = \tau'$ donc $\tau' f'' = \tau' f' = 0_{C''}$. Deux cas sont possibles pour les cellules qui ont σ' dans leur image par f :

▷ $\sigma' \in \tau\partial$, dans ce cas $\sigma' \in \sigma f$ et est remplacée par $\sigma' f'$, c'est-à-dire que $\sigma f'' = \sigma f - \sigma' + \sigma' f'$;

▷ $\sigma' \notin \tau\partial$, dans ce cas seule σ' a σ' dans son image par f, en particulier $\sigma' f = \sigma'$.

Au bout du compte, on observe que selon le couple (τ', σ') réduit, le calcul de f'' par composition consiste parfois à **remplacer** une partie de l'image de σ par f ;

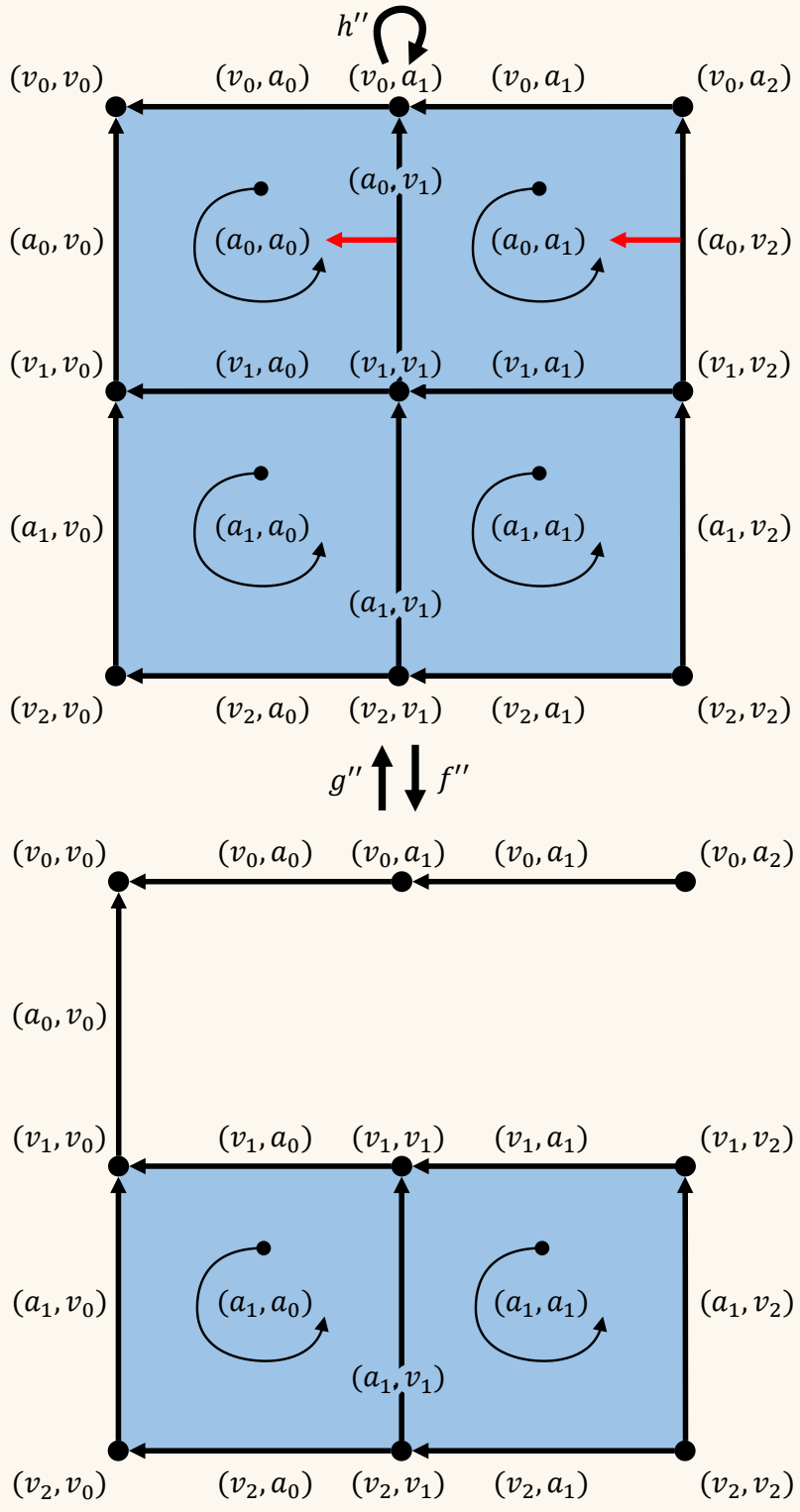


FIGURE 19 – Une réduction $\rho'' : (C, \partial) \Rrightarrow (C'', \partial'')$ calculée par composition de deux réductions élémentaires $\rho : (C, \partial) \Rrightarrow (C', \partial')$, telle que $(a_0, v_2)h = (a_0, a_1)$ (illustrée sur la figure 18) et $\rho' : (C', \partial') \Rrightarrow (C'', \partial'')$, telle que $(a_0, v_2)h' = (a_0, a_0)$ (non illustrée).

— pour n'importe quelle cellule de (C'', ∂''), g'' est l'identité sauf pour :
  · l'étoile directe de σ dans (C, ∂) en dehors de τ. Soit τ'' une telle cellule, $\tau''g'' = \tau'' - \tau''\partial h = \tau'' \pm \tau$;
  · l'étoile directe de σ' dans (C', ∂') en dehors de τ'. Soit τ'' une telle cellule, $\tau''g'' = \tau''g' - \tau''\partial' h' = \tau''g' \pm \tau'$;

  Au bout du compte, on observe que selon les couples (τ, σ) et (τ', σ') réduits, le calcul de g'' par composition consiste parfois à **ajouter** τ et/ou τ' à l'image de l'étoile directe de σ dans (C, ∂) ou de l'étoile directe de σ' dans (C', ∂') ;

— pour n'importe quelle cellule de (C, ∂), h'' est nul sauf pour σ et σ'. Intéressons-nous à ces dernières :
  · par définition, h est nul pour toute cellule sauf $\sigma h = \epsilon\tau$. Lorsqu'on calcule $h'' = h + fh'g$, on ajoute $fh'g$ à toutes les cellules qui ont σ' dans leur image par f car σ' est la seule cellule telle que $\sigma' h' \neq 0$. De cette façon, on construit des **chemins de réduction** au fil des compositions ;
  · les cellules ayant σ' dans leur image par f sont σ', car $\sigma' f = \sigma'$, et σ si $\sigma' \in \tau\partial$, car $\sigma f = (-\epsilon\tau\partial + \sigma)\phi^{-1}$. Dit autrement, σ a σ' dans son image par f si σ' a τ dans son étoile directe dans (C, ∂) ;
  · le nombre de cellules ajoutées aux cellules ayant σ' dans leur image par f dépend de la taille de $\tau' g$ car $\sigma' h' = \epsilon\tau'$;
  · or, $\tau' g = \tau' - \tau'\partial h$, c'est-à-dire qu'on ajoute au minimum τ, auquel s'ajoute la taille de $\tau'\partial h$. En l'occurrence, on ajoute au plus une cellule car h est nul sauf pour $\sigma h = \epsilon\tau$.

  Au bout du compte, on observe que selon les couples (τ, σ) et (τ', σ') réduits, le calcul de h'' par composition consiste parfois à **ajouter** τ à l'image de σ'

Exemple. Considérons l'exemple de la figure 19, qui illustre la composition de deux réductions $\rho : (C, \partial) \Rightarrow (C', \partial')$ caractérisée par (a_0, v_2) et (a_0, a_1), et $\rho' : (C', \partial') \Rightarrow (C'', \partial'')$ caractérisée par (a_0, v_1) et (a_0, a_0). La réduction ρ est illustrée sur la figure 18. Observons que :

— pour n'importe quelle cellule de (C, ∂), f'' est l'identité sauf pour $(a_0, a_1)f'' = (a_0, a_0)f'' = 0_{C''}$ et pour (a_0, v_2) et (a_0, v_1). Intéressons-nous à ces dernières :
  · dans ρ, on a $(a_0, v_2)f = (-(a_0, a_1)\partial + (a_0, v_2)) = -((a_0, v_2) + (v_0, a_1) - (a_0, v_1) - (v_1, a_1)) + (a_0, v_2) = -(v_0, a_1) + (a_0, v_1) + (v_1, a_1)$;
  · dans ρ', on a $(a_0, v_1)f' = -((a_0, v_1) + (v_0, a_0) - (a_0, v_0) - (v_1, a_0)) + (a_0, v_1) = -(v_0, a_0) + (a_0, v_0) + (v_1, a_0)$;
  · f'' est calculée via la formule $f'' = ff'$. Or, f' est l'identité pour toute cellule sauf $(a_0, a_0)f' = 0_{C''}$ et $(a_0, v_1)f' = -(v_0, a_0) + (a_0, v_0) + (v_1, a_0)$. Les cellules pour lesquelles $f'' \neq f'$ sont donc toutes les cellules qui ont (a_0, a_0) ou (a_0, v_1) dans leur image par f ;
  · f est l'identité pour toutes les cellules sauf pour $(a_0, a_1)f = 0_{C'}$ et $(a_0, v_2)f = -(v_0, a_1) + (a_0, v_1) + (v_1, a_1)$. La cellule qui a (a_0, a_0) dans son image par f est (a_0, a_0), donc $(a_0, a_0)f'' = (a_0, a_0)f' = 0_{C''}$. Deux cas sont possibles pour les cellules qui ont (a_0, v_1) dans leur image par f :
    ▷ $(a_0, v_1) \in (a_0, a_1)\partial$, c'est le cas ici et on a $(a_0, v_1)f = (a_0, v_1)$ et $(a_0, v_2)f = -(v_0, a_1) + (a_0, v_1) + (v_1, a_1)$. Composer f et f' revient à remplacer (a_0, v_1) dans $(a_0, v_2)f$ par $(a_0, v_1)f'$, on obtient $(a_0, v_2)f'' = -(v_0, a_1) - (v_0, a_0) + (a_0, v_0) + (v_1, a_0) + (v_1, a_1)$.

▷ $(a_0, v_1) \notin (a_0, a_1)\partial$, ce n'est pas le cas ici.

— pour n'importe quelle cellule de (C'', ∂''), g'' est l'identité car il n'y aucune cellule dans l'étoile directe de (a_0, v_2) dans (C, ∂) en dehors de (a_0, a_1), et il n'y a aucune cellule dans l'étoile directe de (a_0, v_1) par (C', ∂') en dehors de (a_0, a_0) ;

— pour n'importe quelle cellule de (C, ∂), h'' est nul sauf pour (a_0, v_2) et (a_0, a_0). Intéressons-nous à ces dernières :

· par définition, h est nul pour toute cellule sauf $(a_0, v_2)h = (a_0, a_1)$;

· les cellules ayant (a_0, v_1) dans leur image par f sont (a_0, v_1) et (a_0, v_2) car $(a_0, v_1) \in (a_0, a_1)\partial$;

· le nombre de cellules ajoutées aux cellules ayant (a_0, v_1) dans leur image par f dépend de la taille de $(a_0, a_0)g$ car $(a_0, v_1)h' = (a_0, a_0)$;

· or, $(a_0, a_0)g = (a_0, a_0)$;

Au bout du compte, on ajoute donc (a_0, a_0) à l'image de (a_0, v_2) et (a_0, v_1), c'est-à-dire que $(a_0, v_2)h = (a_0, a_1) + (a_0, a_0)$ et $(a_0, v_1)h = (a_0, a_0)$. En particulier, le chemin de réduction de (a_0, v_2) est contient deux 2-cubes après composition.

Réduction optimale. On constate donc que plusieurs critères peuvent faire varier le calcul d'une composition de réduction, dont notamment l'étoile directe et le bord des cellules que l'on choisit de réduire. En conséquence, nous nous interrogeons sur l'existence d'une réduction "optimale", c'est-à-dire d'une réduction qui minimise le nombre de valeurs explicites que contiendront ses matrices ∂, f, g, h et ∂', et la possibilité de la calculer en partant d'un complexe de chaînes. Cette question reste ouverte à ce jour. Nous pensons que faire le lien entre toutes les matrices qui compose une réduction permettrait de répondre à cette question, c'est-à-dire, intuitivement, établir un lien entre ce qui est "gagné" au niveau des opérateurs de bord, et ce qui est "perdu" au niveau des matrices f, g et h lorsqu'on réduit. Les cas extrêmes sont la réduction identité, où la totalité des valeurs explicites sont dans $\partial' = \partial$ (car $f = g = id$ et $h = 0$), et la réduction dans laquelle tout est réduit, où la totalité des valeurs explicites sont dans h et ∂ (car $\partial' = f = g = 0$).

Exemple. La figure 20 illustre deux réductions ρ^A et ρ^B qui consiste à réduire une $(1, e)$-grille en un sommet. La réduction $\rho^A : (C^A, \partial^A) \Rrightarrow (C'^A, \partial'^A)$ est calculée par composition de deux réductions élémentaires, où :

— v_0 est réduit avec a_0 ;

— v_2 est réduit avec $-a_1$.

La réduction $\rho^B : (C^B, \partial^B) \Rrightarrow (C'^B, \partial'^B)$ est calculée par composition de deux réductions élémentaires, où :

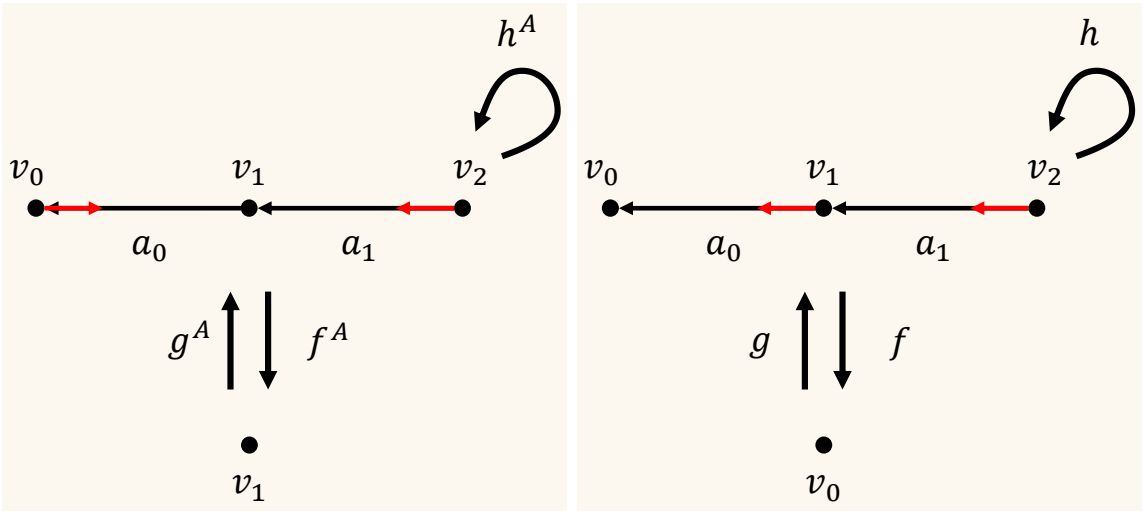
— v_2 est réduit avec $-a_1$;

— v_1 est réduit avec a_0.

En particulier, les matrices $h^{A,1} : C^{A,0} \to C^{A,1}$ et $h^{B,1} : C^{B,0} \to C^{B,1}$ sont :

$$\begin{array}{c} h^{A,1} \\ v_0 \\ v_1 \\ v_2 \end{array} \begin{array}{c} \begin{array}{cc} a_0 & a_1 \end{array} \\ \begin{bmatrix} 1 & . \\ . & . \\ . & -1 \end{bmatrix} \end{array} \qquad \begin{array}{c} h^{B,1} \\ v_0 \\ v_1 \\ v_2 \end{array} \begin{array}{c} \begin{array}{cc} a_0 & a_1 \end{array} \\ \begin{bmatrix} . & . \\ -1 & . \\ -1 & -1 \end{bmatrix} \end{array}$$

Notons que $h_\Delta^{A,1} = 2$ et que $h_\Delta^{B,1} = 3$.



(a) (b)

FIGURE 20 – Deux réductions calculées par composition de réductions élémentaires. Une flèche rouge allant de σ vers σ' indique que ce couple de cellules est réduit dans une réduction élémentaire. (a) $\rho^A : (C^A, \partial^A) \Rrightarrow (C'^A, \partial'^A)$. (b) $\rho^B : (C^B, \partial^B) \Rrightarrow (C'^B, \partial'^B)$.

D Processus de construction : couture de deux grilles

Cette annexe propose une étude approfondie du processus de construction ayant servi à mener l'étude expérimentale de la sous-section 2.4.1. Pour rappel, une (d, e)-grille est l'ensemble semi-cubique qui résulte du produit cartésien de d arêtes subdivisées en e 1-cubes et $1 + e$ sommets.

D.1 Caractéristiques d'une grille

Nommons $G^{k,e}$ le nombre de cubes d'une (k, e)-grille et $G^{k,e}_p$ le nombre de p-cubes d'une (k, e)-grille. On a :

$$G^{k,e} = (2e+1)^d \qquad G^{k,e}_p = \binom{k}{p} e^p (e+1)^{k-p} \qquad \forall k, e, p \in \mathbb{N} | 0 \leq p \leq d$$

Démonstration. Rappelons que le cardinal d'un ensemble résultant d'un produit cartésien est le produit des cardinaux de ces ensembles. Il y a $(2e+1)$ cubes dans chaque $(1, e)$-grille faisant partie du produit cartésien à l'origine d'une (k, e)-grille, ce qui suffit à démontrer $G^{k,e}$. Démontrons $G^{k,e}_p$:

- un p-cube résulte nécessairement du produit de p 1-cubes de par la définition d'une (k, e)-grille. Il y a $\binom{k}{p}$ façons de choisir ces p 1-cubes parmi les d $(1, e)$-grilles ;
- dans chaque $(1, e)$-grille, il y a e 1-cubes et $1 + e$ 0-cubes, d'où $G^{k,e}_p = \binom{k}{p} e^p (e+1)^{k-p}$.

Soit (C, ∂) un complexe de chaînes induit par une (k, e)-grille, donc où $|B_C| = G^{k,e}$. Nommons $G^{k,e}_\Delta$ le nombre de valeurs explicites de ∂ et $G^{k,e}_{p,\Delta}$ le nombre de valeurs explicites de ∂^p. On a :

$$G^{k,e}_\Delta = \sum_{p=1}^{k} 2pG^{k,e}_p \qquad G^{k,e}_{p,\Delta} = 2pG^{k,e}_p \qquad \forall k, e, p \in \mathbb{N} | 0 \leq p \leq k$$

Démonstration. Pour démontrer $G^{k,e}_{p,\Delta}$, il suffit simplement d'analyser la définition d'un ensemble semi-cubique (cf. définition 1.2.11). Il y est dit que pour un ensemble semi-cubique (K, d) de dimension k, d est tel que $(d^p_{i,j})^{1 \leq p \leq k}_{0 \leq j \leq 1 \leq i \leq p}$. Il suffit de compter le nombre de valeurs que prennent i et j :

- j prend 2 valeurs, 0 et 1 ;
- i prend p valeurs, comprises entre 1 et k inclus.

Il y a donc $2p$ $(p-1)$-cubes dans le bord de tout p-cube, d'où $G^{k,e}_{p,\Delta} = 2pG^{k,e}_p$ et $G^{k,e}_\Delta$ en découle.

D.2 Réduction d'une grille

Algorithme. L'algorithme que nous utilisons pour réduire une grille par composition de réductions élémentaires est illustré sur la figure 21. Il sert notamment pour le calcul de $\rho^{S,t=0}$ de l'équivalence homologique $\gamma^{t=0}$ de la couture de deux grilles.

Algorithme : Réduire une (k, e)-grille.

Donnée d'entrée : une (k, e)-grille (C, ∂)
Donnée de sortie : une réduction $\rho : (C, \partial) \Rrightarrow (C', \partial')$

$\rho \leftarrow$ réduction identité ;
pour p *de* k *à* 1 *inclus* **faire**
  $pattern \leftarrow (v_e^0, \ldots, v_e^{k-p}$ ou $a_{e-1}^{k-p}, \ldots, a_{e-1}^{k-1})$;
  $x \leftarrow pattern$;
  $reduireEnDimP \leftarrow vrai$;
  tant que $reduireEnDimP$ **faire**
    $a_i^r \leftarrow$ l'arête la plus à droite dans x;
    $y \leftarrow$ une copie de x, dans laquelle a_i^r est remplacée par v_{i+1}^r;
    Appliquer la réduction élémentaire $\rho' : (C', \partial') \Rrightarrow (C'', \partial'')$ de x et y, et la composer dans $\rho : (C, \partial) \Rrightarrow (C', \partial')$ (donc $(C', \partial') = (C'', \partial'')$ après composition);
    $decrement \leftarrow faux$;
    $q \leftarrow r$;
    tant que $q >= 0$ *et* $non(decrement)$ **faire**
      $\sigma_i^q \leftarrow$ la q-ème composante de x;
      si $i > 0$ **alors**
        Remplacer σ_i^q par σ_{i-1}^q dans x;
        $decrement \leftarrow vrai$;
      sinon
        Remplacer σ_i^q dans x par a_{e-1}^q si σ est une arête et par v_e^q sinon ;
      fin
      $q \leftarrow q - 1$;
    fin
    si $non(decrement)$ **alors**
      $reduireEnDimP \leftarrow$ y a-t-il une arête a_{e-1}^l qui suit un sommet v_e^{l-1} dans $pattern$? ;
      si $reduireEnDimP$ **alors**
        Remplacer a_{e-1}^l dans $pattern$ par v_0^l si $l = r$, et par v_e^l sinon;
        Remplacer v_e^{l-1} dans $pattern$ par a_{e-1}^{l-1};
        $x \leftarrow pattern$;
      fin
    fin
  fin
fin
retourner ρ

FIGURE 21 – Algorithme de réduction d'une $(k - e)$-grille par composition de réductions élémentaires (x, y), où x est un p-cube et y un $(p - 1)$-cube, et pour p compris entre k et 1 inclus.

Le principe de cet algorithme repose sur l'association d'un mot à chaque cube d'une (k, e)-grille représentant les cubes du produit cartésien des k $(1, e)$-grilles dont il émane.

Exemple. Pour une $(2, 2)$-grille, le mot associé au 0-cube qui résulte du produit des sommets v_2 avec v_2 est (v_2, v_2). La figure 22 illustre la réduction d'une $(2, 2)$-grille.

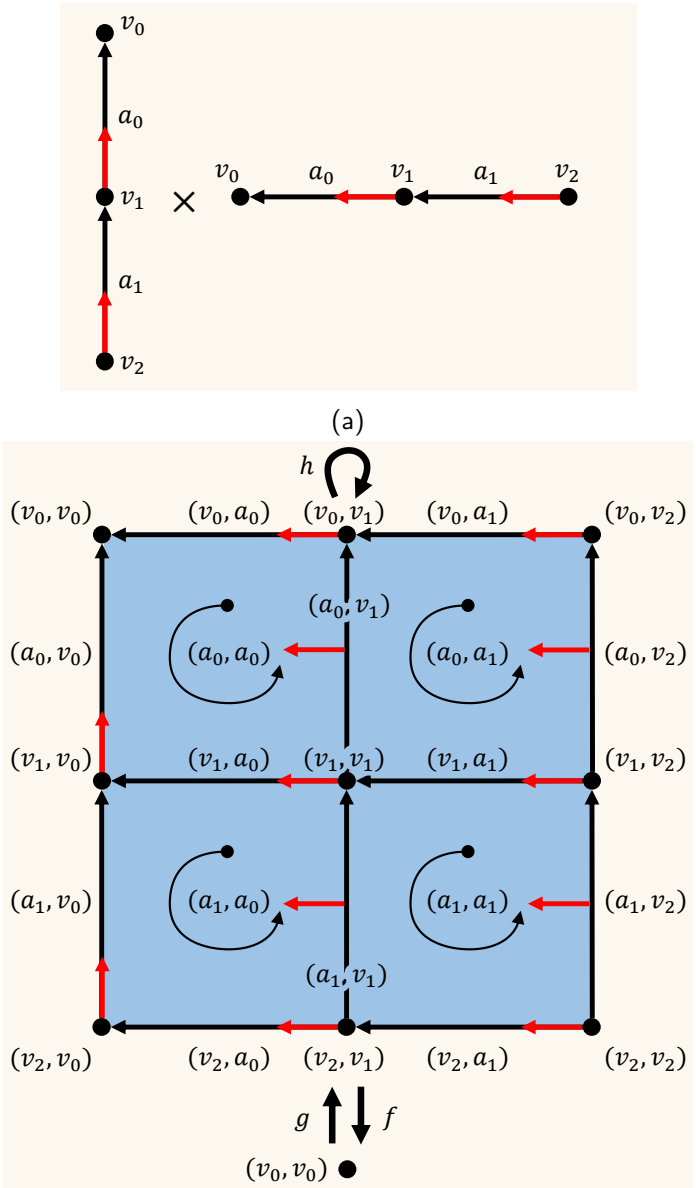


FIGURE 22 – Réduction $\rho : (C, \partial) \Rrightarrow (C', \partial')$ d'une $(2, 2)$-grille sur un sommet. Une flèche rouge allant de σ vers σ' indique que ce couple de cellules est réduit dans une réduction élémentaire. (a) La réduction illustrée sur le produit cartésien. (b) La réduction illustrée sur la $(2, 2)$-grille.

Pour chaque dimension p d'une (k, e)-grille, en partant de $p = k$ et jusqu'à $p = 1$, on détermine un ensemble de couples (x, y) à réduire où x est un p-cube et y un $(p - 1)$-cube. De cette façon, on compose des réductions élémentaires (x, y) jusqu'à ce que tous les p-cubes

soient réduits, puis on passe à la dimension $p-1$ et ainsi de suite jusqu'à ce que la grille réduite soit un sommet. En particulier, pour une valeur de p donnée, le premier p-cube x considéré est $(v_e^0, \ldots, v_e^{k-p}$ ou $a_{e-1}^{k-p}, \ldots, a_{e-1}^{k-1})$, c'est-à-dire celui qui résulte du produit cartésien des $k-p$ sommets v_e choisis parmi les $k-p$ premières $(1,e)$-grilles, avec p arêtes a_{e-1}, choisies parmi les p dernières $(1,e)$-grilles. Le $(p-1)$-cube y a exactement le même mot que x sauf pour sa dernière arête a_i^r, qui est remplacée par v_{i+1}^r.

Exemple. La figure 23 illustre les premiers p-cubes x réduits d'une $(3,2)$-grille pour p de 3 à 1 inclus. En particulier :

— pour $p=3$, il s'agit de (a_1, a_1, a_1), qui est donc réduit avec (a_1, a_1, v_2) ;

— pour $p=2$, il s'agit de (v_2, a_1, a_1), qui est donc réduit avec (v_2, a_1, v_2) ;

— pour $p=1$, il s'agit de (v_2, v_2, a_1), qui est donc réduite avec (v_2, v_2, v_2).

Considérons le 1-cube (v_2, v_0, a_0). Il est réduit avec (v_2, v_0, v_1) car la dernière arête de son mot est a_0, qui est remplacé par v_1 pour déterminer le $(p-1)$-cube avec qui il est réduit.

Pour un $pattern$ donné, on détermine un ensemble de p-cubes à réduire en décrémentant ses lettres en partant de la droite. Décrémenter une lettre signifie que a_i devient a_{i-1} ou v_i devient v_{i-1} pour $i > 0$. Lorsqu'une lettre d'indice q ne peut pas être décrémentée (car $i = 0$), elle prend la valeur a_{e-1}^q si c'est une arête ou v_e^q si c'est un sommet, et on essaie de décrémenter la lettre $q-1$.

Exemple. Pour une $(3,2)$-grille, le 2-cube qui suit (v_1, a_1, a_1) est (v_1, a_1, a_0). Le 2-cube qui suit (v_1, a_1, a_0) est (v_1, a_0, a_1) etc. . .

Le mot associé au premier p-cube réduit détermine un $pattern$ de dimension p, c'est-à-dire une combinaison indiquant dans quelles $(1,e)$-grilles sont choisis les sommets et les arêtes à l'origine des p-cubes réduits. Il existe $\binom{k}{p}$ $pattern$ pour la dimension p.

Exemple. Pour une $(3,2)$-grille, le premier $pattern$ de réduction des 2-cubes est (v_2, a_1, a_1), qui indique que l'on va réduire tous les 2-cubes qui résulte du produit cartésien d'un sommet avec une arête avec une arête, dans cet ordre. Une fois tous ces cubes réduits (le dernier étant (v_0, a_0, a_0)), le $pattern$ suivant est (a_1, v_2, a_1), qui indique que l'on va réduire tous les 2-cubes qui résulte du produit cartésien d'une arête avec un sommet avec une arête, dans cet ordre (le dernier étant (a_0, v_0, a_0)). Enfin, le dernier $pattern$ est (a_1, a_1, v_0), qui indique que l'on va réduire tous les 2-cubes qui résulte du produit cartésien d'une arête avec une arête avec le sommet v_0, dans cet ordre (le dernier étant (a_0, a_0, v_0)). Observons que la dernière lettre du $pattern$ (a_1, a_1, v_0) est v_0 car les 2-cubes comme (a_1, a_1, v_2) sont déjà réduits avec des 3-cubes, (a_1, a_1, a_1) en l'occurrence pour (a_1, a_1, v_2).

Tous les p-cubes issus d'un $pattern$ sont réduits lorsqu'il n'est plus possible de décrémenter de lettres. Dans ce cas, on passe au $pattern$ suivant (l.22 de l'algorithme) en cherchant s'il existe une lettre a_{e-1}^l qui suit une lettre v_e^{l-1} dans le $pattern$. Puis :

— si c'est le cas, le $pattern$ suivant est calculé en échangeant les lettres a_{e-1}^l et v_e^{l-1} qui deviennent respectivement,

  · a_{e-1}^{l-1} et v_e^l dans le $pattern$ si l ne correspond pas à l'indice de l'arête la plus à droite dans $pattern$;

  · a_{e-1}^{l-1} et v_0^l sinon. Cette conditionnelle sert à prendre en compte le fait que certains p-cubes soient réduits avec des $(p+1)$-cubes, et ne sont donc pas réduits avec des $(p-1)$-cubes. En particulier, de par la définition de x et y, on sait que ces p-cubes

se terminent forcément par un sommet différent de v_0 car le p-cube y a le même mot que le $(p+1)$-cube x sauf pour l'arête la plus à droite a_i^r du mot de x, qui devient v_{i+1}^r ;

— si ce n'est pas le cas, alors on a terminé de traiter la dimension p et on passe à la dimension $p-1$.

Exemple. Considérons une $(3,2)$-grille, on a dans l'ordre :

— les réductions élémentaires des 3-cubes (a_1,a_1,a_1), (a_1,a_1,a_0), (a_1,a_0,a_1), (a_1,a_0,a_0), (a_0,a_1,a_1), (a_0,a_1,a_0), (a_0,a_0,a_1), (a_0,a_0,a_0) ;

— les réductions élémentaires des 2-cubes (v_2,a_1,a_1), (v_2,a_1,a_0), (v_2,a_0,a_1), (v_2,a_0,a_0), (v_1,a_1,a_1), (v_1,a_1,a_0), (v_1,a_0,a_1), (v_1,a_0,a_0), (v_0,a_1,a_1), (v_0,a_1,a_0), (v_0,a_0,a_1), (v_0,a_0,a_0), (a_1,v_2,a_1), (a_1,v_2,a_0), (a_1,v_1,a_1), (a_1,v_1,a_0), (a_1,v_0,a_1), (a_1,v_0,v_0), (a_0,v_2,a_1), (a_0,v_2,a_0), (a_0,v_1,a_1), (a_0,v_1,a_0), (a_0,v_0,a_1), (a_0,v_0,a_0), (a_1,a_1,v_0), (a_1,a_0,v_0), (a_0,a_1,v_0), (a_0,a_0,v_0). En particulier, le passage dans la branche de la l.22 de l'algorithme se fait au moment où :

· $x=(v_0,a_0,a_0)$ et $pattern=(v_2,a_1,a_1)$, qui devient alors $x=pattern=(a_1,v_2,a_1)$;

· $x=(a_0,v_0,a_0)$ et $pattern=(a_1,v_2,a_1)$ qui devient alors $x=pattern=(a_1,a_1,v_0)$;

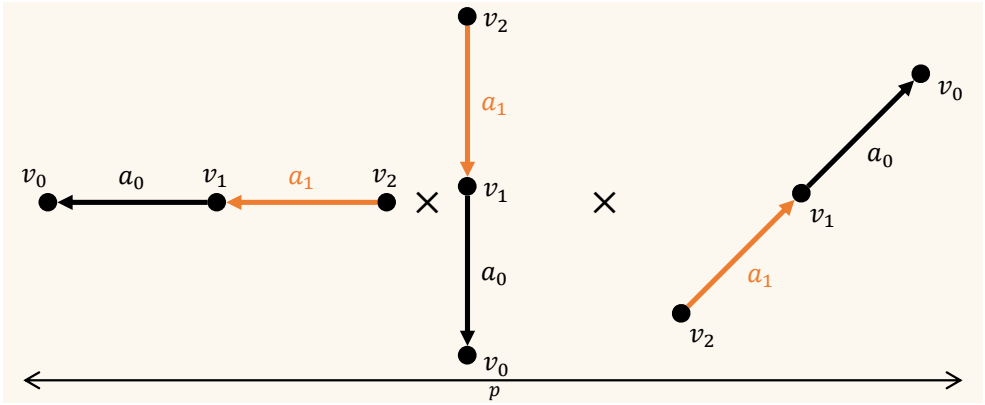
· la boucle tant que de la l.6 s'arrête pour $x=(a_0,a_0,v_0)$ et $pattern=(a_1,a_1,v_0)$.

— les réductions élémentaires des 1-cubes (v_2,v_2,a_1), (v_2,v_2,a_0), (v_2,v_1,a_1), (v_2,v_1,a_0), (v_2,v_0,a_1), (v_2,v_0,a_0), (v_1,v_2,a_1), (v_1,v_2,a_0), (v_1,v_1,a_1), (v_1,v_1,a_0), (v_1,v_0,a_1), (v_1,v_0,a_0), (v_0,v_2,a_1), (v_0,v_2,a_0), (v_0,v_1,a_1), (v_0,v_1,a_0), (v_0,v_0,a_1), (v_0,v_0,a_0), (v_2,a_1,v_0), (v_2,a_0,v_0), (v_1,a_1,v_0), (v_1,a_0,v_0), (v_0,a_1,v_0), (v_0,a_0,v_0), (a_1,v_0,v_0), (a_0,v_0,v_0). En particulier, le passage dans la branche de la l.22 de l'algorithme se fait au moment où :

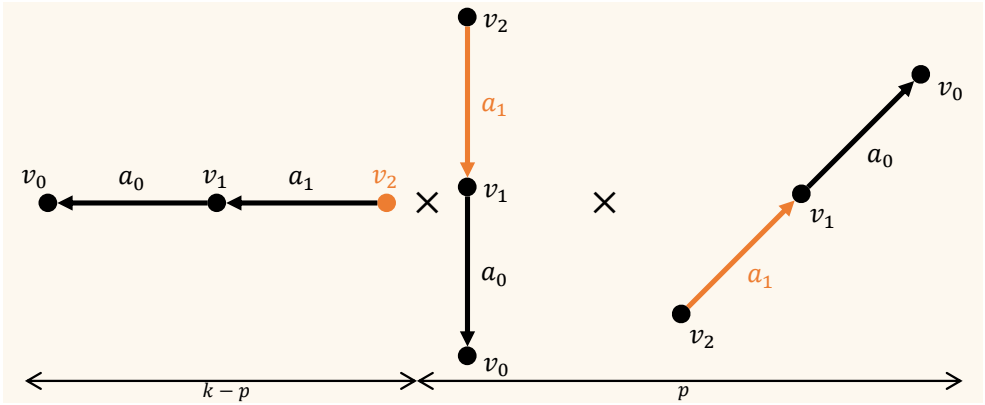
· $x=(v_0,v_0,a_0)$ et $pattern=(v_2,v_2,a_1)$, qui devient alors $x=pattern=(v_2,a_1,v_0)$;

· $x=(v_0,a_0,v_0)$ et $pattern=(v_2,a_1,v_0)$ qui devient alors $x=pattern=(a_1,v_0,v_0)$;

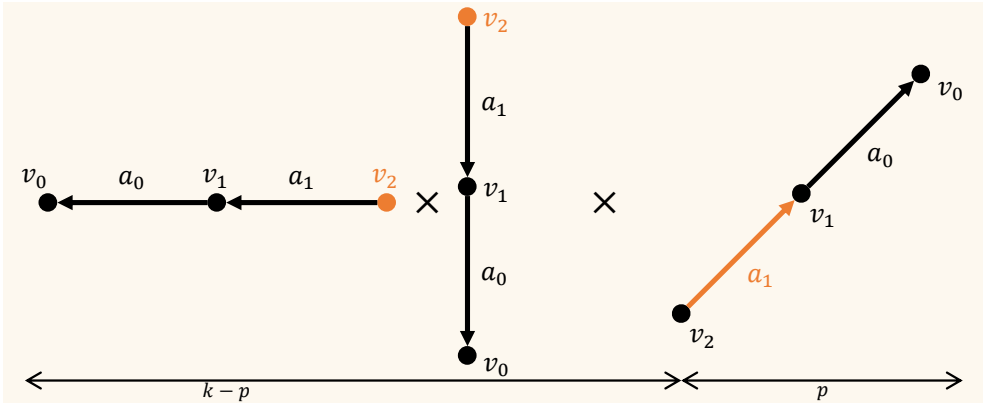
· la boucle tant que de la l.6 s'arrête pour $x=(a_0,v_0,v_0)$ et $pattern=(a_1,v_0,v_0)$.



(a) $p = 3$



(b) $p = 2$



(c) $p = 1$

FIGURE 23 – Les premiers p-cubes réduits dans une $(3, 2)$-grille pour différentes valeurs de p. (a) Pour $p = 3$, le premier cube est (a_1, a_1, a_1). (b) Pour $p = 2$, (v_2, a_1, a_1). (c) Pour $p = 1$, (v_2, v_2, a_1).

M	Nb. de lignes	Nb. de colonnes	Variation	M_Δ
∂	$G^{k,e}$	$G^{k,e}$	$[0]$	$2pG_p^{k,e}$
f	$G^{k,e}$	1	$[1]$	$G_0^{k,e}-1$
g	1	$G_p^{k,e}$	$[1]$	0
h	$G^{k,e}$	$G^{k,e}$	$[0]$	$\sum_{p=0}^{k-1}\binom{k}{p+1}e^p(1+e)^{k-1-p}\frac{e(e+1)}{2}$
∂'	1	1	$[0]$	0

TABLE 1 – Les caractéristiques des matrices de la réduction d'une (k,e)-grille calculée avec l'algorithme de la figure 21.

Caractéristiques de la réduction. Soit $\rho : (C,\partial) \Rightarrow (C',\partial')$ la réduction d'une (k,e)-grille calculée avec l'algorithme précédent. Les caractéristiques de ses matrices sont répertoriées dans le tableau 1. La colonne M_Δ indique le nombre de valeurs explicites des matrices, en supposant que l'identification et les valeurs nulles sont représentées implicitement.

Démonstration. Nous démontrons les formules de la colonne M_Δ du tableau 1 à l'exception de la formule de ∂_Δ, déjà démontrée dans l'annexe D.1 :

- ∂_Δ découle du fait que toute grille se réduit sur un sommet. Le bord d'un sommet étant nul, on a $\partial_\Delta = 0$;
- f_Δ associe l'ensemble des sommets de la grille au sommet qui résulte de la réduction. Pour l'un d'entre eux, il s'agit d'une identité, d'où $f_\Delta = G_0^{k,e} - 1$;
- g_Δ associe l'unique sommet qui résulte de la réduction à "lui-même" dans la (k,e)-grille. Il s'agit donc d'une identité, d'où $g_\Delta = 0$;
- h_Δ se décompose en plusieurs parties :
  - $\frac{e(e+1)}{2}$ représente le nombre de valeurs explicites dans h pour la réduction d'une $(1,e)$-grille. La formule consiste à dénombrer le nombre de fois où $\frac{e(e+1)}{2}$ se retrouve dans la réduction d'une (k,e)-grille ;
  - $\binom{k}{p+1}$ représente le nombre de $pattern$ de dimension $p+1$;
  - pour un $pattern$ donné, la réduction d'une $(1,e)$-grille se retrouve dans le fait de décrémenter une composante qui représente un sommet en partant de v_{e+1} ;
  - il y a $e^p(1+e)^{k-1-p}$ p-cubes qui peuvent être calculés en excluant cette composante ;
  - en appliquant ce raisonnement à tous les p-cubes ayant une image par h, donc de $p=0$ jusqu'à $p=k-1$ inclus, on retrouve $h_\Delta = \sum_{p=0}^{k-1}\binom{k}{p+1}e^p(1+e)^{k-1-p}\frac{e(e+1)}{2}$;

Exemple. Illustrons la démonstration de h_Δ sur la réduction d'une $(3,2)$-grille. Le nombre de valeurs explicites dans h pour la réduction d'une $(1,2)$-grille est $(2*3)/2 = 3$ car :

- $v_2h = -a_1 - a_0$ et $v_1h = -a_0$.

Dénombrons le nombre de fois où la réduction d'une $(1,e)$-grille se retrouve dans la réduction d'une (k,e)-grille pour $p=2$. Il y a $\binom{3}{3} = 1$ $pattern$ de dimension 3, (a_1,a_1,a_1), réduit sur sa dernière composante. Il y a $2^2 * (2+1)^{3-1-2} = 4$ façons de calculer des 2-cubes en bloquant une composante et en ayant une image par h :

- $(a_1,a_1,v_2)h = -(a_1,a_1,a_1) - (a_1,a_1,a_0)$ et $(a_1,a_1,v_1)h = -(a_1,a_1,a_0)$;
- $(a_1,a_0,v_2)h = -(a_1,a_0,a_1) - (a_1,a_0,a_0)$ et $(a_1,a_0,v_1)h = -(a_1,a_0,a_0)$;

— $(a_0, a_1, v_2)h = -(a_0, a_1, a_1) - (a_0, a_1, a_0)$ et $(a_0, a_1, v_1)h = -(a_0, a_1, a_0)$;
— $(a_0, a_0, v_2)h = -(a_0, a_0, a_1) - (a_0, a_0, a_0)$ et $(a_0, a_0, v_1)h = -(a_0, a_0, a_0)$.

Dénombrons le nombre de fois où la réduction d'une $(1, e)$-grille se retrouve dans la réduction d'une (k, e)-grille pour $p = 1$. Il y a $\binom{3}{2} = 3$ $patterns$ de dimension 2, (v_2, a_1, a_1), (a_1, v_2, a_1) et (a_1, a_1, v_0). Pour chacun de ces patterns, il y a $2^1(2+1)^{3-1-1} = 6$ façons de calculer des 1-cubes en bloquant une composante et en ayant une image par h. Considérons le pattern (v_2, a_1, a_1), on a :

— $(v_2, a_1, v_2)h = -(v_2, a_1, a_1) - (v_2, a_1, a_0)$ et $(v_2, a_1, v_1)h = -(v_2, a_1, a_0)$;
— $(v_2, a_0, v_2)h = -(v_2, a_0, a_1) - (v_2, a_0, a_0)$ et $(v_2, a_0, v_1)h = -(v_2, a_0, a_0)$;
— $(v_1, a_1, v_2)h = -(v_1, a_1, a_1) - (v_1, a_1, a_0)$ et $(v_1, a_1, v_1)h = -(v_1, a_1, a_0)$;
— $(v_1, a_0, v_2)h = -(v_1, a_0, a_1) - (v_1, a_0, a_0)$ et $(v_1, a_0, v_1)h = -(v_1, a_0, a_0)$;
— $(v_0, a_1, v_2)h = -(v_0, a_1, a_1) - (v_0, a_1, a_0)$ et $(v_0, a_1, v_1)h = -(v_0, a_1, a_0)$;
— $(v_0, a_0, v_2)h = -(v_0, a_0, a_1) - (v_0, a_0, a_0)$ et $(v_0, a_0, v_1)h = -(v_0, a_0, a_0)$;

Considérons le pattern (a_1, v_2, a_1), on a :

— $(a_1, v_2, v_2)h = -(a_1, v_2, a_1) - (a_1, v_2, a_0)$ et $(a_1, v_2, v_1)h = -(a_1, v_2, a_0)$;
— $(a_1, v_1, v_2)h = -(a_1, v_1, a_1) - (a_1, v_1, a_0)$ et $(a_1, v_1, v_1)h = -(a_1, v_1, a_0)$;
— $(a_1, v_0, v_2)h = -(a_1, v_0, a_1) - (a_1, v_0, a_0)$ et $(a_1, v_0, v_1)h = -(a_1, v_0, a_0)$;
— $(a_0, v_2, v_2)h = -(a_0, v_2, a_1) - (a_0, v_2, a_0)$ et $(a_0, v_2, v_1)h = -(a_0, v_2, a_0)$;
— $(a_0, v_1, v_2)h = -(a_0, v_1, a_1) - (a_0, v_1, a_0)$ et $(a_0, v_1, v_1)h = -(a_0, v_1, a_0)$;
— $(a_0, v_0, v_2)h = -(a_0, v_0, a_1) - (a_0, v_0, a_0)$ et $(a_0, v_0, v_1)h = -(a_0, v_0, a_0)$;

Considérons le pattern (a_1, a_1, v_0), on a :

— $(a_1, v_2, v_2)h = -(a_1, a_1, v_0) - (a_1, a_0, v_0)$ et $(a_1, v_1, v_2)h = -(a_1, a_0, v_0)$;
— $(a_1, v_2, v_1)h = -(a_1, a_1, v_0) - (a_1, a_0, v_0)$ et $(a_1, v_1, v_1)h = -(a_1, a_0, v_0)$;
— $(a_1, v_2, v_0)h = -(a_1, a_1, v_0) - (a_1, a_0, v_0)$ et $(a_1, v_1, v_0)h = -(a_1, a_0, v_0)$;
— $(a_0, v_2, v_2)h = -(a_0, a_1, v_0) - (a_0, a_0, v_0)$ et $(a_0, v_1, v_2)h = -(a_0, a_0, v_0)$;
— $(a_0, v_2, v_1)h = -(a_0, a_1, v_0) - (a_0, a_0, v_0)$ et $(a_0, v_1, v_1)h = -(a_0, a_0, v_0)$;
— $(a_0, v_2, v_0)h = -(a_0, a_1, v_0) - (a_0, a_0, v_0)$ et $(a_0, v_1, v_0)h = -(a_0, a_0, v_0)$;

Il en est de même pour $p = 0$.

D.3 Caractéristiques d'un n-motif

Un n-motif est une partie du produit cartésien à l'origine d'une (k, e)-grille. Intuitivement, il est construit en choisissant une arête a_i et le sommet de son bord v_{i+1} dans les n premières $(1, e)$-grilles du produit, et le sommet v_0 dans les $k - n$ dernières $(1, e)$-grilles, pour i compris entre 0 et e inclus.

Exemple. La figure 24 illustre les cubes choisis pour construire un 2-motif dans une $(3, 2)$-grille.

Dans la couture de deux grilles, à partir de l'étape 1 de construction, les complexes de chaînes (C, ∂) de chaque SECE

$$(C, \partial) \underset{\mathsf{r}}{\overset{\mathsf{i}}{\rightleftarrows}} (C^t, \partial^t) \underset{\mathsf{s}}{\overset{\mathsf{j}}{\rightleftarrows}} (C^{t+1}, \partial^{t+1})$$

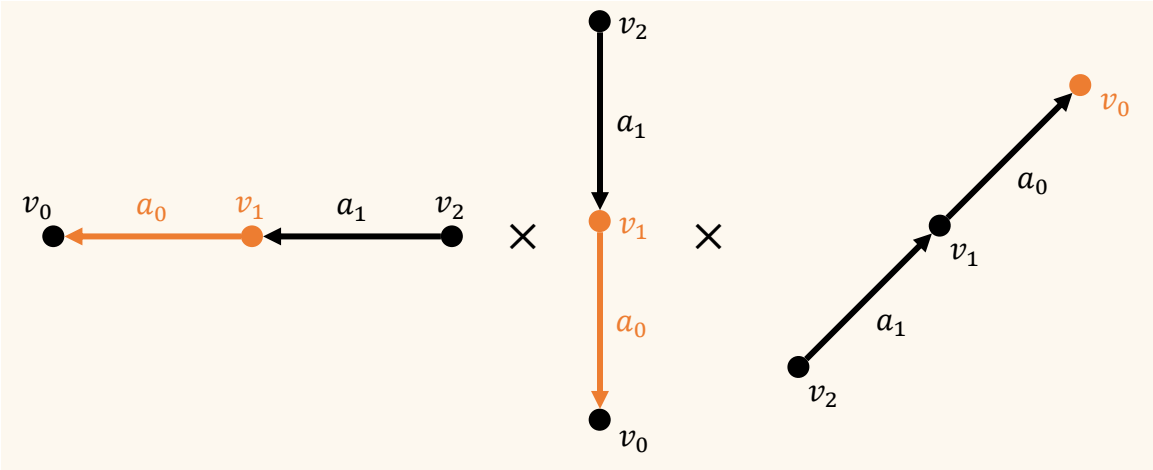


FIGURE 24 – Les cubes choisis pour construire un 2-motif dans une $(3,2)$-grille.

sont induits par un n-motif, les complexes de chaînes (C^t, ∂^t) sont induits par les deux grilles avant identification, et les complexes de chaînes $(C^{t+1}, \partial^{t+1})$ sont induits par les deux grilles après identification. Nommons T^p le nombre de p-cubes d'un n-motif, T son nombre de cubes (donc $|B_C| = T$), T^p_Δ le nombre de $(p-1)$-cubes dans le bord de ses p-cubes, et T_Δ le nombre de valeurs explicites induites par le bord de ses cubes (donc $\partial_\Delta = T_\Delta$). On a :

$$T^p = \binom{n}{p} \qquad T = 2^n \qquad T^p_\Delta = \binom{n}{p} p \qquad T_\Delta = \sum_{p=1}^{n} \binom{n}{p} p$$

Démonstration. La démonstration de T^p est triviale : un p-cube du motif résulte du produit de p arêtes choisies parmi les n premières $(1,e)$-grille, d'où $T^p = \binom{n}{p}$. La démonstration de T en découle, car $T = \sum_{p=0}^{n} \binom{n}{p}$ et, par application de la formule du binôme de Newton, $T = 2^n$. La démonstration de T^p_Δ s'appuie sur le fait qu'un n-motif est construit en choisissant strictement un sommet v_{i+1} dans le bord de chaque arête a_i. Restreignons le produit cartésien à l'origine d'un motif aux n premières $(1,e)$-grilles : un p-cube du motif résulte du produit de p arêtes a_i avec $n-p$ sommets v_{i+1}. Dit autrement, les $(p-1)$-cubes dans le bord d'un p-cube donné sont obtenus en remplaçant une de ces arêtes a_i par un sommet v_{i+1}. Il y a p façons de choisir cette arête, et il y a $\binom{n}{p}$ p-cubes, d'où $T^p_\Delta = \binom{n}{p} p$.

Exemple. Nous illustrons la démonstration de T^p_Δ sur l'exemple de la figure 24. Considérons :

- la dimension 2. Il y a $\binom{2}{2} = 1$ 2-cube dans le 2-motif, (a_0, a_0, v_0). Il y a 2 façons de choisir un sommet parmi ses 2 arêtes : (a_0, v_1, v_0) et (v_1, a_0, v_0). D'où $T^2_\Delta = \binom{2}{2} 2 = 2$;
- la dimension 1. Il y a $\binom{2}{1} = 2$ 1-cubes dans le 2-motif, (a_0, v_1, v_0) et (v_1, a_0, v_0). Il y a 1 façon de choisir un sommet parmi la seule arête de ces 1-cubes : (v_1, v_1, v_0). D'où $T^1_\Delta = \binom{2}{1} 1 = 2$.

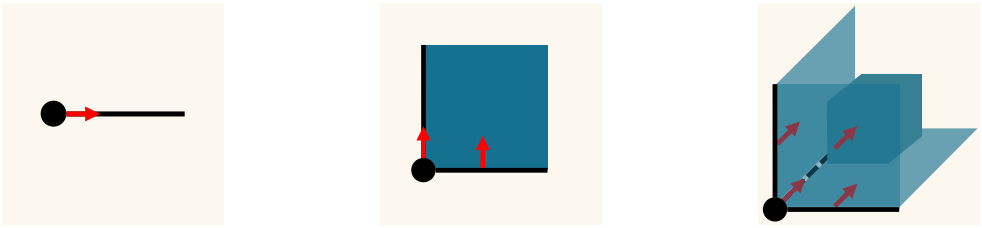
D.4 Réduction d'un n-motif

Un 1-motif est une arête a_i et un sommet v_{i+1} tels que v_{i+1} apparaît dans le bord de a_i avec un coefficient -1. Le couple (a_i, v_{i+1}) caractérise une réduction élémentaire, et donc un

1-motif se réduit sur le vide. Par extension, tout n-motif se réduit sur le vide. En particulier, il existe une réduction dans laquelle tout chemin de réduction a au plus une taille de 1, c'est-à-dire que le nombre de valeurs explicites de h pour cette réduction correspond au nombre de p-cubes pour p compris entre 0 et $n-1$ inclus. La figure 25 illustre ce paragraphe. Cette réduction est ρ^S dans toutes les équivalences homologiques

$$\gamma : (C, \partial) \overset{\rho}{\Leftarrow} (C^B, \partial^B) \overset{\rho^S}{\Rightarrow} (C^S, \partial^S)$$

utilisées dans l'application du théorème SECE aux identifications de la couture de deux grilles, à partir de l'étape 1. Ses caractéristiques sont répertoriées dans le tableau 2.



(a) $n = 1$ (b) $n = 2$ (c) $n = 3$

FIGURE 25 – La réduction de n-motifs pour différentes valeurs de n, et dans laquelle tout chemin de réduction a au plus une taille de 1.

M	Nb. de lignes	Nb. de colonnes	Variation	M_Δ
∂	T	T	$[0]$	T_Δ
f	T	0	$[1]$	0
g	0	T	$[1]$	0
h	T	T	$[0]$	$\sum_{p=0}^{n-1} T^p$
∂'	0	0	$[0]$	0

TABLE 2 – Caractéristiques de la réduction d'un n-motif.

E La dégénérescence induit une SECE

Dans cette section, nous montrons que la SECE S,

$$(C, \partial) \underset{\text{r}}{\overset{\text{i}}{\rightleftarrows}} (C^t, \partial^t) \underset{\text{s}}{\overset{\text{j}}{\rightleftarrows}} (C^{t+1}, \partial^{t+1})$$

définie dans le paragraphe 4.3.4-a est une SECE :

— (C, ∂) est un complexe de chaînes car $\partial = 0$ donc $\partial^2 = 0$;

— Montrons que i est un morphisme de complexe de chaînes, c'est-à-dire qu'il vérifie $\partial i = i\partial^t$;

- · les générateurs σ induits par les simplexes dégénérés engendrent C et $\sigma i = \sigma$ pour tout $\sigma \in C$;
- · or, $\sigma\partial^t = 0_{C^t}$, donc $i\partial^t = 0_{C^t}$;
- · $\partial = 0_C$ donc $\partial i = 0_{C^t}$;
- · $\partial i = i\partial^t = 0_{C^t}$, donc i est un morphisme de complexe de chaînes.

— Montrons que j est un morphisme de complexe de chaînes, c'est-à-dire qu'il vérifie $\partial^t j = j\partial^{t+1}$:

- · si σ est un simplexe dégénéré, alors :
  - ▷ $\sigma j = 0_{C^{t+1}}$ donc $\sigma j\partial^{t+1} = 0_{C^{t+1}}$;
  - ▷ $\sigma\partial^t = 0_{C^t}$ donc $\sigma\partial^t j = 0_{C^{t+1}}$;
  - ▷ d'où $\sigma j\partial^{t+1} = \sigma\partial^t j = 0_{C^{t+1}}$;
- · si σ est un simplexe non dégénéré, alors :
  - ▷ $\sigma j = \sigma$ et $\sigma j\partial^{t+1}$ revient à envoyer sur $0_{C^{t+1}}$ les simplexes dégénérés de $\sigma\partial^t$;
  - ▷ $\sigma\partial^t j$ revient à envoyer les simplexes de $\sigma\partial^t$ sur $0_{C^{t+1}}$;
  - ▷ d'où $\sigma j\partial^{t+1} = \sigma\partial^t j$.
- · donc j est un morphisme de complexe de chaînes.

— La démonstration de $ir = id_C$ est triviale car :

- · C est engendré par les simplexes dégénérés ;
- · $\sigma i = \sigma$ pour tout $\sigma \in C$;
- · $\sigma r = \sigma$ pour les générateurs de C^t induit par un simplexe dégénéré.

— La démonstration de $sj = id_{C^{t+1}}$ est triviale car :

- · par définition, C^{t+1} est engendré par les simplexes non dégénérés ;
- · $\sigma s = \sigma$ pour tout $\sigma \in C^{t+1}$;
- · $\sigma j = \sigma$ pour les générateurs de C^t induit par un simplexe non dégénéré.

— Montrons que $ri + js = id_{C^t}$. Pour cela, considérons un générateur σ de C^t induit par un simplexe :

- · dégénéré. Dans ce cas, $\sigma ri = \sigma$ et $\sigma js = 0_{C^t}$, donc $\sigma(ri + js) = \sigma$;
- · non dégénéré. Dans ce cas, $\sigma ri = 0_{C^t}$ et $\sigma js = \sigma$, donc $\sigma(ri + js) = \sigma$;

On en déduit que $ri + js = id_{C^t}$.

F La composition de deux SECE relevant du même cas du théorème SECE est une SECE

Soit

$$S'' : (C'', \partial'') \underset{\mathsf{r''}}{\overset{\mathsf{i''}}{\rightleftarrows}} (C^t, \partial^t) \underset{\mathsf{s''}}{\overset{\mathsf{j''}}{\rightleftarrows}} (C^{t+2}, \partial^{t+2})$$

une SECE qui résulte de la composition de deux SECE relevant du même cas du théorème SECE (cf. paragraphe 4.3.5-a)

$$S : (C, \partial) \underset{\mathsf{r}}{\overset{\mathsf{i}}{\rightleftarrows}} (C^t, \partial^t) \underset{\mathsf{s}}{\overset{\mathsf{j}}{\rightleftarrows}} (C^{t+1}, \partial^{t+1})$$

et

$$S' : (C', \partial') \underset{\text{r'}}{\overset{\text{i'}}{\rightleftarrows}} (C^{t+1}, \partial^{t+1}) \underset{\text{s'}}{\overset{\text{j'}}{\rightleftarrows}} (C^{t+2}, \partial^{t+2})$$

Montrons que S'' est une suite exacte courte effective :

— (C'', ∂'') est un complexe de chaînes car

$$\begin{aligned}
(\partial'')^2 &= (\partial + \partial' + \delta)^2 \\
&= \partial^2 + \partial\partial' + \partial\delta + \partial'\partial + (\partial')^2 + \partial'\delta + \delta\partial + \delta\partial' + \delta^2 \\
&= 0 + 0 + 0 + 0 + 0 + \partial'\delta + \delta\partial + 0 + 0 \\
&= \partial'(i's\partial^t r) + (i's\partial^t r)\partial \\
&= i'\partial^{t+1}s\partial^t r + i's\partial^t r\partial \qquad \text{car } \partial' i' = i'\partial^{t+1} \\
&= i'(\partial^{t+1}s\partial^t r + s\partial^t r\partial) \\
&= 0 \text{ car } -\partial^{t+1}s\partial^t r = s\partial^t r\partial \text{ d'après [16, p.10]}
\end{aligned}$$

— i'' est un morphisme de complexe de chaînes car

$$\begin{aligned}
\partial'' i'' &= (\partial + \partial' + \delta)(i + i's) \\
&= \partial i + \partial i's + \partial' i + \partial' i's + \delta i + \delta i's \\
&= \partial i + 0 + 0 + \partial' i's + \delta i + 0 \\
&= i\partial^t + i'\partial^{t+1}s + \delta i \text{ car } \partial i = i\partial^{t+1} \text{ et } \partial' i' = i'\partial^{t+1} \\
&= i\partial^t + i'\partial^{t+1}s + (i's\partial^t r)i \\
&= i\partial^t + i'(\partial^{t+1}s + s\partial^t ri) \\
&= i\partial^t + i'(s\partial^t) \\
&= (i + i's)\partial^t \\
&= i''\partial^t
\end{aligned}$$

en utilisant le fait que

$$\begin{aligned}
\partial^{t+1}s + s\partial^t ri &= \partial^{t+1}s + s\partial^t(id_{C^t} - js) \\
&= \partial^{t+1}s + s\partial^t - s\partial^t js \\
&= \partial^{t+1}s + s\partial^t - sj\partial^{t+1}s \\
&= s\partial^t \text{ car } sj = id_{C^{t+1}}
\end{aligned}$$

— j'' est un morphisme de complexes de chaînes car

$$\begin{aligned}
\partial^t j'' &= \partial^t jj' \\
&= jj'\partial^{t+2} \text{ car } \partial^t j = j\partial^{t+1} \text{ et } \partial^{t+1}j' = j'\partial^{t+1}
\end{aligned}$$

— $i''r'' = id_{C''}$ car

$$\begin{aligned} i''r'' &= (i + i's)(r + jr') \\ &= ir + ijr' + i'sr + i'sjr' \\ &= id_C + 0 + 0 + i'sjr \text{ car } ij = 0 \text{ et } sr = 0 \\ &= id_C + i'id_{C^{t+1}}r \text{ car } sj = id_{C^{t+1}} \\ &= id_C + id_{C'} \text{ car } i'r' = id_{C'} \\ &= id_{C''} \end{aligned}$$

— $s''j'' = id_{C^{t+2}}$ car

$$\begin{aligned} s''j'' &= s'sjj' \\ &= s'id_{C^{t+1}}j' \text{ car } sj = id_{C^{t+1}} \\ &= id_{C^{t+2}} \text{ car } s'j' = id_{C^{t+1}} \end{aligned}$$

— $r''i'' + j''s'' = id_{C^t}$ car

$$\begin{aligned} r''i'' + j''s'' &= (r + jr')(i + i's) + jj's's \\ &= ri + ri's + jr'i + jr'i's + jj's's \\ &= ri + 0 + 0 + jr'i's + jj's's \\ &= (id_{C^t} - js) + j(id_{C^{t+1}} - j's')s + jj's's \text{ car } ri = (id_{C^t} - js) \text{ et } r'i' = (id_{C^{t+1}} - j's') \\ &= id_{C^t} - js + js - jj's's + jj's's \\ &= id_{C^t} \end{aligned}$$