UNIVERSITÉ DE CAEN BASSE-NORMANDIE

U.F.R. Sciences

ECOLE DOCTORALE SIMEM

THÈSE

Présentée par

M. Jean-Hugues Pruvot

et soutenue

le 18 novembre 2008

en vue de l'obtention du

DOCTORAT de l'UNIVERSITÉ DE CAEN BASSE-NORMANDIE
Spécialité : Informatique

(Arrêté du 6 août 2006)

Segmentation et appariement hiérarchiques basés sur les pyramides combinatoires

MEMBRES du JURY

M. Jacques-Olivier Lachaud, Professeur, Université de Savoie ..... Président du Jury
M. Luc Brun, Professeur, ENSICAEN ..... Directeur de Thèse
M. Jean-Philippe Domenger, Professeur, Université de Bordeaux ..... Rapporteur
Mme. Aline Deruyver, Maître de conférence (HDR), Université de Strasbourg . Rapporteur
M. Abderrahim Elmoataz, Professeur, Université de Caen ..... Examinateur

TABLE DES MATIÈRES

TABLE DES FIGURES

INTRODUCTION

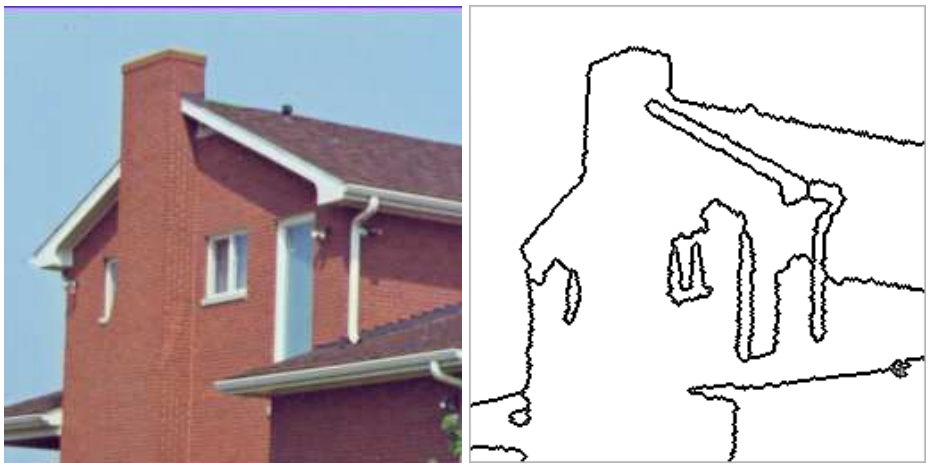
NOUS assistons actuellement à une explosion du «tout numérique» qui amène à devoir stocker de plus en plus de documents électroniques de tous types, qu'il s'agisse d'écrits, d'images, de musiques ou de séquences vidéo. Pour pouvoir s'y retrouver dans ces masses considérables de données, des méthodes d'indexation et de classification sont nécessaires. Dans le cadre du traitement d'images, une étape préliminaire à bon nombre d'applications d'indexation de classification mais également de suivi ou de reconnaissance d'objets passent par un processus permettant de mettre en évidence un objet ou une structure commune à plusieurs images.

Il existe de nombreuses approches permettant de résoudre ce type de problème. Une des approches les plus prometteuses dans ce domaine est la mise en correspondance structurelle. Les méthodes d'appariement structurel sont en général effectuées sur des points d'intérêt ou une représentation de l'image en régions. Les méthodes d'appariement de points ont été abondemment étudiées essentiellement en raison de la simplicité et de la robustesse de ses descripteurs. Toutefois les informations de région décrivent des zones de l'image et portent donc beaucoup plus d'information. Dans la suite de ce document nous allons proposer des méthodes de construction de partitionnement d'images en région et des méthodes d'appariement des régions obtenues.

La segmentation peut être définie comme un processus de synthèse, visant à extraire des caractéristiques géométriques des images en faisant abstraction des nuances de couleurs, des textures, des transparences et autres reflets qui la composent. Comment aborder un tel problème ? Prenons par exemple la maison présentée sur la figure 1.1. Cette information sémantique peut être représentée par la segmentation présentée sur la figure 1.1 (b) où la forme de la maison se détache du fond. Est-ce uniquement parce que nous connaissons le concept de maison que nous somme capable d'en tracer les contours ?

De nombreux travaux de psychologues ayant étudié la vision humaine dans le courant du XXéme siècle, et notamment l'école du Gelstalt, tendent à prouver que notre perception visuelle aurait d'avantage tendance à effectuer le raisonnement dans l'autre sens (voir les travaux de Palmer [PALMER99] et Gordon [GORDON97] pour une synthèse de ces travaux), et nous serions ainsi capable de reconnaitre un objet parce que nous savons en détecter les contours. L'hypothèse formulée par ces chercheurs est qu'un traitement bas niveau agirait lors des stades initiaux de l'acquisition des stimuli visuels, dans l'unique but de structurer le flot de données brutes perçues par la rétine. L'extraction des caractéristiques géométriques de l'image permettrait dans un second temps la réalisation de tâches de plus haut niveau, telles que la catégorisation ou la reconnaissance.

La démarche méthodologique que nous allons suivre pour aborder le problème d'appariement s'inspire de ces idées. Nous allons dans un premier temps segmenter l'image



(a) (b)

FIG. 1.1 – Exemple de segmentation d'une image naturelle

en tenant compte uniquement de ses données physiques, sans connaître préalablement son contenu. Nous utiliserons ensuite la partition fournie par le processus de segmentation pour mettre en œuvre la mise en correspondance

Le problème de la segmentation est néanmoins un problème mal posé, essentiellement du fait que la notion d'objet elle-même n'est pas complètement définie. En effet lorsqu'un être humain observe une image naturelle, il y voit généralement des objets physiques ou leurs parties. Il est donc capable de diviser cette image en parties, ou en *segments* les représentant. Les éléments composant l'image vont alors disposer de plusieurs niveaux de description. Si l'on observe par exemple une maison, la bonne segmentation est-elle celle proposant les contours extérieurs de la maison, ou une partition décomposant la maison en différents éléments tels que les fenêtres, la porte, le garage. Toutes ces descriptions sont finalement valides et nous ne pouvons affirmer arbitrairement qu'une description proposant un niveau de finesse plus ou moins important est meilleure ou moins bonne qu'une autre.

Pour pallier à cette ambiguïté, nous proposons d'utiliser une segmentation hiérarchique d'une image permettant d'obtenir des segmentations présentant différents niveaux de détails. Pour segmenter une image naturelle, un humain va identifier différents objets physiques et démarquer leur contours jusqu'à un certain niveau de détail, determiné par l'attention qu'il leur accorde. La segmentation hiérarchique tente de reproduire ce processus, réalisé naturellement par notre cortex cérébral. Pour obtenir une segmentation robuste et visuellement cohérente des objets d'intérêt, présent dans les images, nous allons incorporer dans le processus de construction des hiérarchies, différents critères énergétiques permettant, par exemple, d'obtenir une modélisation de l'image comme une fonction continue par morceau.

Il existe différentes familles de méthodes permettant d'approcher le problème d'appariement de partitions. La complexité des méthodes d'appariement structurelles est en général très importante et elle nécessite souvent l'utilisation d'heuristiques pour espérer approcher la solution optimale. Nombre d'entre elles se basent sur une structure de graphe représentant les relations entre les régions et transforment le problème en un problème d'appariement de graphe. L'un des inconvénients majeurs de ce type d'approche est dû

au problème d'inconsistance inhérent à toute méthode de segmentation. Il est en effet très rare d'obtenir une correspondance exacte entre une région et un véritable objet. De fait, un objet se retrouve le plus souvent, défini par l'union de plusieurs régions ou inversement une région peut très bien englober plusieurs objets.

Le fait d'utiliser des hiérarchies nous permet d'outrepasser le problème d'inconsistance en proposant plusieurs niveaux de détail pour chaque image à apparier. Nous allons de plus utiliser les pyramides combinatoires pour représenter nos hiérarchies. Ce type de structure possède des propriétés intéressantes qui vont nous permettre de guider le processus d'appariement et faire baisser la complexité du problème. Elle fournit

1. un codage naturel de l'orientation du plan,
2. un vecteur de caractéristiques associé à chaque région,
3. le voisinage de chaque région, représenté à différents niveaux pour affiner le processus de mise en correspondance.

Les pyramides combinatoires on également l'avantage de ne nécessiter que très peu d'espace mémoire pour être stockées et manipulées.

Le manuscrit s'articule en deux parties. Dans la première partie nous allons étudier la segmentation hiérarchique au travers de différentes heuristiques permettant de construire des hiérarchies de partitions. Dans un premier temps, dans le chapitre 2, nous allons rappeler quelques définitions ainsi que les principales propriétés des différentes structures hiérarchiques. Nous proposerons ensuite dans le chapitre 3 différentes heuristiques permettant la construction de hiérarchies de partitions. Ces approches, basées sur la théorie ensemble-échelle proposée par Guigues, vont nous permettre d'obtenir un compromis entre le temps d'exécution et l'énergie des partitions obtenues.

La seconde partie considère l'appariement de régions dans deux hiérarchies. Nous proposons, dans le chapitre 4 un tour d'horizon des principales approches pour l'appariement de graphes planaires : les approches algorithmiques et les approches par optimisation. Nous présenterons finalement dans le chapitre 5 une nouvelle approche au problème de l'appariement de régions. Notre approche utilise les propriétés topologiques, géométriques, photométrique mais également hiérarchiques, fournis par les pyramides combinatoires, pour mettre en correspondance deux régions issues de deux hiérarchies de segmentations. Nous obtenons finalement l"appariement de deux régions ainsi que la mise en correspondance d'éléments de frontières définis à différents niveaux dans la hiérarchie. La mise en correspondance de ces éléments de frontière correspond à l'appariement de deux voisinages de région dans l'ensemble des coupes des deux hiérarchies.

PREMIÈRE PARTIE

SEGMENTATION HIERARCHIQUE

REPRÉSENTATION HIÉRARCHIQUE

Sommaire

DANS ce chapitre, nous allons rappeler les définitions ainsi que les principales propriétés des structures qui nous seront utiles par la suite. Par structure nous entendons d'une part les structures mathématiques (abstraites) utiles pour modéliser notre problème mais également les structures informatiques utiles pour la manipulation de segmentations

2.1 Graphe

Pour la terminologie classique de théorie des graphes, nous renvoyons le lecteur à un ouvrage de référence tel que " Graphes " de C.Berge [BERGE70, BERGE74, BERGE83]. Nous rappelons ici brièvement les quelques éléments nécessaires à la bonne compréhension du manuscrit dans le cadre du codage et de la manipulation de partitions.

Cette première section, étant principalement une suite de définition, elle s'en trouve naturellement aride et un lecteur familiarisé avec les notions de graphe pourra la parcourir relativement vite.

Définition 1 Graphe

Un graphe $G = (V, E)$ est la donnée d'un ensemble fini V appelé ensemble des sommets et d'un sous-ensemble E de $V \times V$, appelé ensemble des arêtes de G.

Une arête $e \in E$ est définie par une paire non-ordonnée de sommets, appelés les extrémités de e. Deux sommets a et b définissant une arête $e = (a, b)$ sont dit adjacents. L'arête e est dite incidente à a et b.

Sauf mention contraire tous les graphes considérés dans ce document seront non orientés. Une arête reliant deux sommets a et b pourra donc être indifféremment notée (a, b) ou (b, a).

Définition 2 Boucle

Dans un graphe $G = (V, E)$ une arête $e = (a, a)$ deux fois incidente à un même sommet est appelée une boucle.

Définition 3 Arbre

Un arbre est un graphe connexe ne contenant pas de cycle.

Définition 4 Arbre de recherche

Un arbre de recherche est un arbre dont chacun des nœuds correspond à un état et chaque arête code une transition entre les états qu'elle relie.

Définition 5 Feuille

Dans un arbre de recherche, une feuille est définie comme un état ne pouvant amener à un état différent. Une feuille est donc soit un arbre vide soit une extrémité d'un arbre ayant pour fils des arbres vides.

Définition 6 Forêt

Une forêt est un graphe défini comme l'union d'arbres deux à deux disjoints. C'est donc un graphe acyclique dont chacune des composantes connexe est un arbre.

Définition 7 Graphe simple

Une graphe est dit simple s'il ne contient aucune boucle et si l'adjacence entre deux sommets est codée par au plus une arête.

Définition 8 Graphe partiel

Un graphe $G' = (V, E')$ est appelé un graphe partiel de $G = (V, E)$, si E' est inclus dans E. Autrement dit, on obtient G' en enlevant une ou plusieurs arêtes au graphe G.

Définition 9 graphe labélisé

*Un **graphe labélisé** $G = (V, E, \mu, \nu, L_v, L_e)$ est défini par l'ensemble de ses sommets V, de ses arêtes E et par deux fonctions de label μ et ν :*

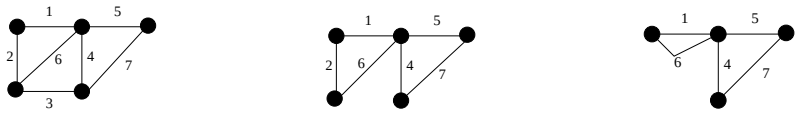
$$\begin{cases} \mu : V \rightarrow L_v & \text{fonction de label sur les sommets} \\ \nu : E \rightarrow L_e & \text{fonction de label sur les arêtes} \end{cases}$$

Définition 10 Sous graphe labélisé

*Un **sous graphe labélisé** $G_s = (V_s, E_s, \mu_s, \nu_s, L_v, L_e)$ d'un graphe $G = (V, E, \mu, \nu, L_v, L_e)$ est défini par μ_s et ν_s, restrictions de μ et ν à $V_s \subset V$ et $E_e \subset E$ (avec $E_s = E \cap V_s \times V_s$).*

Définition 11 Suppression

La suppression d'une arête $e \in E$ d'un graphe $G = (V, E)$ définit le graphe partiel $G' = (V, E - \{e\})$.



(a) G (b) $G \setminus 3$ (c) $G/2$

FIG. 2.1 – Suppression(b) puis contraction(c) d'une arête du graphe (a).

Soit $e = (x, y)$ une arête d'un graphe $G = (V, E)$. On note G/e le graphe obtenu de G par contraction de l'arête e en un nouveau sommet v_e qui devient adjacent à l'ensemble des voisins de x et y. La contraction d'arête se définit formellement de la façon suivante :

Définition 12 Contraction

La contraction d'une arête $e = (x, y) \in E$ avec $x \neq y$ d'un graphe $G = (V, E)$ définit le graphe $G' = V', E')$ défini par :

$$V' = V \cup \{v_e\}$$

où v_e est un nouveau sommet associé à e

$$E' = \{(v, w) \in E | \{v, w\} \cap \{x, y\} = \emptyset\} \cup \{(v_e, w) | (x, w) \in E - \{e\} \text{ ou } (y, w) \in E - \{e\}\}$$

Notons que la suppression d'une arête dans un graphe simple préserve la simplicité du graphe. Ce n'est pas le cas de la contraction qui peut créer des arêtes doubles ou des boucles. Ce phénomène est illustré sur la figure 2.1. Notons également que l'opération de contraction n'est pas définie pour les boucles.

Définition 13 Sous-graphe

Soit un graphe $G = (V, E)$, pour un sous-ensemble de sommets A inclus dans V, le sous-graphe de G induit par A est le graphe $G' = (A, E(A))$ dont l'ensemble des sommets est A et l'ensemble des arêtes $E(A)$ est formé de toutes les arêtes de G ayant leurs deux extrémités dans A. Autrement dit, on obtient G' en enlevant un ou plusieurs sommets au graphe G, ainsi que toutes les arêtes incidentes à ces sommets.

L'objet de ce manuscrit est la création et l'appariement de partitions. Comme nous le verrons dans les sections suivantes, les graphes planaires fournissent un outil naturellement adapté au codages de tels objets. Les définitions suivantes des graphes planaires sont empruntés à [ROY69].

Définition 14 Isomorphisme de graphe

Deux graphe $G = (V, E)$ et $G' = (V', E')$ sont dits isomorphes, si leurs ensembles de sommets et d'arcs se correspondent par les bijections :

$$\begin{array}{lll} v_i \in V & \leftrightarrow & v'_i \in V' \\ v_j \in V & \leftrightarrow & v'_j \in V' \end{array} \quad (v_i, v_j) \in E \Leftrightarrow (v'_i, v'_j) \in E'$$

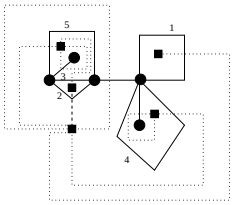


FIG. 2.2 – Un graphe planaire $G = (\bullet, -)$ et son dual $\overline{G} = (\blacksquare, \dots)$ définissant une arête double et une boucle vide.

Définition 15 Graphe planaire

Un graphe planaire topologique est un graphe plongé dans $\mathbb{R}^2$ dont les sommets sont des points distincts, les arêtes des courbes simples et tel que deux arêtes ne se rencontrent pas en dehors de leurs extrémités.

Tout graphe isomorphe à un graphe planaire topologique est un graphe planaire.

Définition 16 Graphe dual

Le graphe dual $\overline{G} = (\overline{V}, \overline{E})$ d'un graphe planaire connexe $G = (V, E)$ est défini de la façon suivante :

- *à toute face f de G on associe un sommet de $\overline{V}$*
- *à toute arête $e \in E$, on associe une arête $\overline{e} \in \overline{E}$ reliant les deux faces f_1 et f_2 séparées par e dans G.*

Le graphe dual de G est donc le graphe dont les sommets correspondent aux faces de G et où deux sommets sont adjacents s'ils correspondent à deux faces adjacentes. On désignera souvent le graphe initial G sous le terme de graphe primal pour le différencier de son dual.

Nous pouvons noter qu'il existe une relation bijective entre les arêtes d'un graphe et celles de son dual. De plus le dual du dual de G est égal au graphe primal :

$$\overline{\overline{G}} = G$$

Notons également qu'il existe une correspondance entre certaines configurations d'arêtes dans G et $\overline{G}$. Par exemple, une boucle dans G est associée à un pont dans $\overline{G}$ alors que réciproquement un pont présent dans le primal est représenté par une boucle dans son dual.

Enfin, la planarité des graphes nous permet de caractériser certains types d'arêtes :

Définition 17 Boucle vide

Soit G un graphe planaire dont le dual est $\overline{G}$. Une boucle vide est une boucle définissant une face de degré 1 (et donc associée à un sommet de degré 1 dans $\overline{G}$).

Définition 18 Arête double

Soit G un graphe planaire dont le dual est $\overline{G}$. Une arête double est une arête contribuant à la définition d'une face de degré 2. (donc associée à un sommet de degré 2 dans $\overline{G}$).

Ces deux derniers types d'arêtes sont illustrés sur la Figure 2.2 où l'arête 1 définit une boucle vide tandis que les arêtes 2 et 3 définissent des arêtes doubles. Notez que 4 ne définit pas une boucle vide tandis que 5 ne définit pas une arête double. En effet les faces associées à ces arêtes ne sont respectivement pas de degré 1 et 2.

Dans le cas du codage de partition avec un graphe planaire, il peut être utile de définir la notion de graphe de régions et de graphe de frontières.

Définition 19 Graphe de Régions

Un graphe planaire G connexe associé à une partition d'une image est appelé un graphe de régions si chaque sommet de G est associé à une région.

Définition 20 Graphe de Frontières

Un graphe planaire G connexe associé à une partition d'une image est appelé un graphe de frontières si chaque sommet de G code l'intersection de frontières de régions.

2.1.1 Définition relatives aux chaînes

Soient Σ un alphabet de symboles, Σ^* l'ensemble des chaînes finies définies sur cet alphabet et $X = x_1 \dots x_n$ et $Y = y_1 \dots y_n$ deux chaînes de tailles respectives n et $m > 0$. La distance entre les chaînes X et Y est définie en termes d'opérations d'édition élémentaires nécessaires à la transformation de X en Y avec un coût minimum. Les trois opérations d'édition de base sont :

1. **substitution** d'un symbole $x \in \Sigma$ par un symbole $y \in \Sigma$, notée $x \to y$
2. **insertion** d'un symbole $x \in \Sigma$, notée $\lambda \to x$
3. **suppression** d'un symbole $y \in \Sigma$, notée $y \to \lambda$

dans lesquelles λ représente la chaîne vide.

Définition 21 Séquence d'édition

*Une **séquence d'édition** S est définie comme une séquence ordonnée d'opérations d'édition élémentaires $s_1, \dots, s_p$.*

Pour deux chaînes structurellement proches, il existe ainsi une séquence d'édition de faible coût alors que pour deux chaînes dont la structure est très différente une séquence d'édition, dont le coût est important, sera nécessaire.

La **distance d'édition** de deux graphes est alors définie comme le chemin d'édition de coût minimum entre ces deux graphes.

Définition 22 Distance d'édition entre chaîne

Soit c une fonction de coût qui attribue à chaque opération d'édition s une valeur réelle positive $c(s)$. Le coût d'un séquence d'édition est donc défini comme la somme des coûts de chacunes des opérations d'édition élémentaires qui la composent :

$$c(S) = \sum_{i=1}^{p} c(s_i) \tag{2.1}$$

*La **distance d'édition** entre deux chaînes X et Y est alors définie comme le coût minimum correspondant à une séquence d'édition permettant de transformer X en Y :*

$$d(X,Y) = \min\{c(S) : \textit{S est une séquence d'opérations d'édition qui transforme } X \textit{ en } Y\} \tag{2.2}$$

Définition 23 Chaînes équivalentes

*Soit $X^{(i)}$ la chaîne ayant subit une rotation circulaire de i positions (i.e $X^{(i)} = x_{i+1} \dots x_n x_1 \dots x_i$). Deux chaînes X et X' de longueur n sont dites **équivalentes** si $X' = X^{(i)}$ pour un entier $1 < i < n$. Ceci définit une relation d'équivalence sur Σ^*.*

Définition 24 Chaîne circulaire

*La classe d'équivalence d'un chaîne X, notée $\overline{X}$, est appelée une **chaîne circulaire**.*

Définition 25 Distance d'édition circulaire

*La **distance d'édition circulaire** entre deux chaînes X et Y est définie par*

$$d_c(X,Y) = \min\{d(X^{(i)}, Y^{(i)}) : i = 1, \dots, n \quad j = 1, \dots, m\} \tag{2.3}$$

La section suivante sera axée sur la représentation de partitions d'images 2D, mais toutes ces notions peuvent s'étendre à des espaces de dimension arbitraire

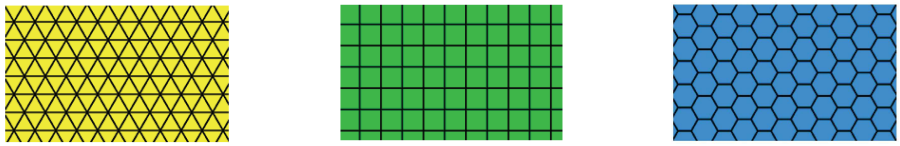
2.2 Image discrète

On désigne sous le terme d'image numérique toute image acquise, créée, traitée ou stockée sous forme binaire. L'étape d'acquisition est réalisée par des convertisseurs analogique-numérique situés dans des dispositifs tels que les scanners, les appareils photo, les caméscopes numériques ou tout dispositif possédant un capteur sensible à la lumière reçue. Ces capteurs ont de nombreuses caractéristiques : leur définition, leur sensibilité spectrale, leur sensibilité en luminosité, leur rémanence,.... Pour obtenir une image correcte et la restituer au mieux et sans perte, il est nécessaire d'organiser ces capteurs de façon à obtenir un pavage du plan.

Il existe de nombreux pavages du plan permettant une discrétisation de l'espace $\mathbb{R}^2$, mais pour représenter les images, les pavages réguliers sont le plus souvent utilisés. Ces pavages vérifient les propriétés suivantes :

- Toutes les cellules sont identiques,
- Toutes les cellules sont convexes,
- Les arêtes définissant les cellules ne se coupent qu'à leurs extrémités,
- les cellules sont régulières (les lignes définissant le contour des cellules sont de longueurs identiques et définissent un angle fixe à leurs intersection)
- Les sommets des cellules sont incidents à un nombre fixe de cellules.

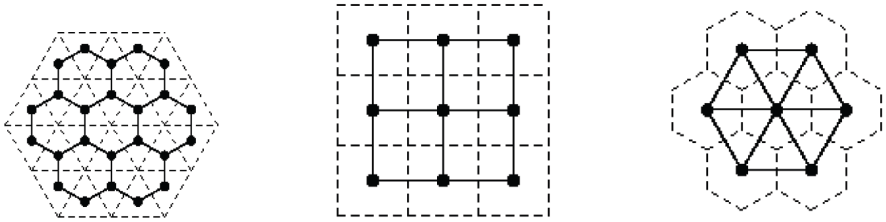
Dans le plan $\mathbb{R}^2$, seuls trois pavages vérifient l'ensemble de ces propriétés : les pavages triangulaires, carrés et hexagonaux (Figure 2.3).



(a) triangulaire (b) carré (c) hexagonal

FIG. 2.3 – Exemples de pavages réguliers du plan

A tout pavage du plan on peut associer un graphe où les sommets représentent les intersections de cellules et les arêtes les relations d'adjacences entre ces sommets (Fig. 2.3). Un tel type de graphe est évidement planaire et son dual (Défs. 15 et 16) est appelé un maillage. Chaque sommet d'un maillage code une cellule du pavage deux sommets du maillage sont adjacents si les cellules correspondantes partagent une arête.



(a) Maillage hexagonal (b) Maillage carré (c) Maillage triangulaire

FIG. 2.4 – Exemples de maillages réguliers du plan

Les images usuelles sont définies sur un maillage carré 2D régulier. Dans la suite de cette section nous limiterons donc nos définitions à ce type de maillage.

L'ensemble des cellules d'un tel maillage peut être représenté par une grille discrète $\{1,\ldots,n\}\times\{1,\ldots,m\}$ où chaque cellule est associée à deux coordonnées entières i et j. Les relations d'ajacences à l'intérieur de la grille discrète sont définies à travers la notion de voisinage.

Définition 26 Voisinage sur une grille discrète

Étant donnée une grille discrète $\{1,\ldots,n\}\times\{1,\ldots,m\}$ *et un sommet* (i,j) *de cette grille, on définit le* 4 *et* 8 *voisinage de* (i,j) *de la façon suivante :*

– 4 *voisinage :*

$$\begin{aligned} V_4(i,j) &= \{(i+1,j),(i-1,j),(i,j),(i,j+1),(i,j-1)\} \\ V_4(i,j) &= \{(i',j')\in\mathbb{N}^2[|i-i'|+|j-j'|=1\} \end{aligned}$$

Notons que le graphe déduit de cette relation de voisinage correspond au maillage. Cela revient à dire que deux cellules sont 4*-voisines si elles ont au moins un coté en commun.*

– *8 voisinage*

$$\begin{aligned} V_8(i,j) &= \cup_{k=-1}^{1}\cup_{l=-1}^{1}\{(i+k,j+l)\} \\ V_8(i,j) &= \{(i',j')\in\mathbb{N}^2[max\{|i-i'|+|j-j'|\}=1\} \end{aligned}$$

Deux cellules sont donc 8-voisines si elles partagent au moins un côté ou si elles ont un sommet en commun.

La figure 2.5(a) et (b) illustre les deux types de voisinages précédemment définis sur un maillage carré. Le graphe défini par les relations V_4 correspond au graphe du maillage.

(a) 4-connexité (b) 8-connexité

FIG. 2.5 – Différents types de connexité rencontrés dans un maillage carré.

La clôture transitive de la relation de voisinage précédemment définie est une relation d'équivalence, notée $\sim$, que l'on peut interpréter comme « il existe un chemin connexe entre x et y »

Définition 27 Chemin

Un chemin est une suite de points, définis sur la grille d'un maillage, tel que chaque point appartient au voisinage du point suivant.

$$x \sim y \Leftrightarrow \exists\{x_1, \ldots, x_n\}/x \in V(x_1), \ldots, x_i \in V(x_{i+1}), \ldots, x_n \in V(y)$$

Le chemin est dit « fermé » si le dernier point est égal au premier point. On peut noter que cette notion de chemin est relative au maillage et à la connexité choisie. On peut donc introduire la notion de x-chemin, comme étant un chemin défini en x-connexité.

Définition 28 Région

Un ensemble de sommets X_i d'une image sera dit connexe si et seulement si pour tout couple de points (P, Q) de X_i il existe un chemin contenu dans X_i et joignant P et Q

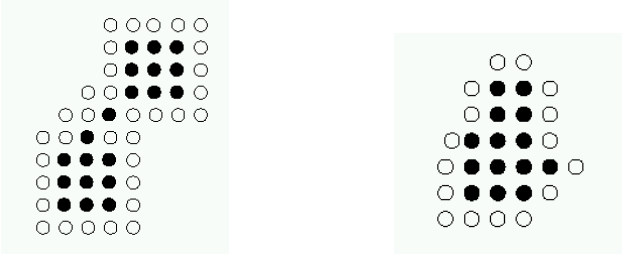
$$\forall (P,Q) \in X_i^2 \quad \exists P = P_1, \ldots, P_n = Q | \forall_i \in \{2, \ldots, n\} \quad P_i \in X_i \ \textit{et} \ P_i \in V(P_{i-1})$$

.

Un ensemble de pixel x-connexe est appelé une région x connexe.

Notons, qu' avec les maillages carrés, il est parfois nécessaire de considérer des choix opposés d'adjacence et de connexité pour l'objet (la figure) et son complémentaire (le fond). Si on utilise par exemple la 4-connexité pour représenter une figure, il faudra prendre la 8-connexité pour le fond, et inversement (figure 2.6).

La notion de **région** définie en image, et utilisée dans la suite du manuscrit, correspondra donc à un ensemble connexe de pixels. Maintenant que cette notion est clairement définie nous allons pouvoir définir une partition.



(a) 8-4connexité (b) 4-8connexité

FIG. 2.6 – Différents types de connexités pour un chemin fermé.

Définition 29 Partition

Pour les images numériques 2D s'appuyant sur un maillage carré, dont le domaine est une partie de $\mathbb{Z}^2$, une partition est définie par la donnée d'un ensemble de régions couvrant l'ensemble de l'image et disjointes 2 à 2.

$$P = \{R_1 \dots R_2\} \quad avec \quad \forall_{i \neq j} R_j \cup R_j = 0 \quad et \quad P = \bigsqcup_{i=1}^{n} R_i$$

De nombreux travaux ont été réalisés autour de la notion de contour dans une image discrète[KOVALE89]. Il en ressort que pour pouvoir définir correctement les contours d'une partition, de façon à ce qu'ils vérifient les propriétés topologiques habituelles (comme le théorème de Jordan [ALEXAN61], l'utilisation d'une topologie basée sur la notion d'interpixel apparaît comme une bonne solution.

Cette notion d'interpixel revient à expliciter les différentes cellules intervenant dans un pavage ou un maillage du plan. Ces différentes cellules sont présentées sur la figure 2.7, dans le cas d'un image discrète définie en deux dimensions.

- Les cellules de dimension 2 sont appelées les pixels
- Les cellules de dimension 1 sont appelées les lignels
- Les cellules de dimension 0 sont appelées les pointels

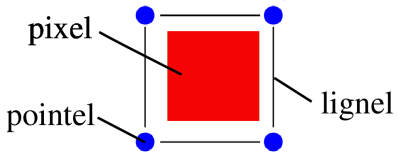


FIG. 2.7 – Les différents éléments interpixelaires en dimension 2.

Ces cellules sont liées les unes aux autres par la relation d'incidence. Deux cellules sont dites incidentes si l'une appartient au bord de l'autre. Elles sont dites adjacentes si elles sont de même dimension et toutes deux incidentes à une même cellule de dimension (n-1).

Grâce à ces notions nous pouvons à présent définir formellement les notions de courbe et de frontière.

Définition 30 Frontières d'une partition

Courbe interpixels : *Une courbe est définie comme une suite de pointels et de lignels* $c_0, \ldots, c_{2k}$ *vérifiant :*
- $\forall_i \in \{0, \ldots, k\}, c_{2i}$ *est un pointel*
- $\forall_i \in \{0, \ldots, k-1\}, c_{2i+1}$ *est un lignel*
- $\forall_i \in \{0, \ldots, 2k-1\}, c_i et c_{i+1}$ *sont incidents*

Lignel frontière : *Dans le cadre d'une partition, un lignel frontière est un lignel bordé par des pixels appartenant à deux régions différentes.*

Courbe frontière : *Une courbe frontière est une courbe interpixel dont tous les lignels sont des lignels frontière.*

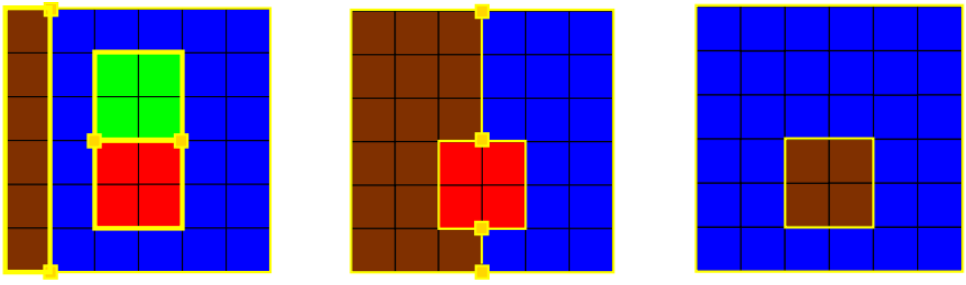
Une courbe est dite simple quand elle ne s'auto-intersecte pas ou encore si toutes ses cellules sont distinctes. Elle est dite fermée si $c_0 = c_{2k}$.

Nous pouvons donc définir la frontière interpixel entre deux régions R_1 et R_2 comme un ensemble de courbes frontières simples. Nous appelerons **nœud** tout pointel d'une frontière interpixel, relié à au moins 3 lignels-frontière et **segment** toute suite maximum de lignels frontières située entre deux nœuds.

Définition 31 Adjacence de région

On dira de deux régions qu'elles sont adjacentes, si elles sont bordées par une même courbe-frontière.

Cette définition de l'adjacence est une relation binaire. Notons que la notion d'adjacence peut recouvrir des notions très diverses. En effet, les régions bleu et marron de la Figure 2.8 pourront avoir une seule frontière commune (Figure 2.8(a)) ou bien avoir plusieurs frontières communes (Figure 2.8(b)) ou encore être incluses l'une dans l'autre (Figure 2.8(c)). Toutes ces configurations correspondront à la même notion d'adjacence entre les deux régions.



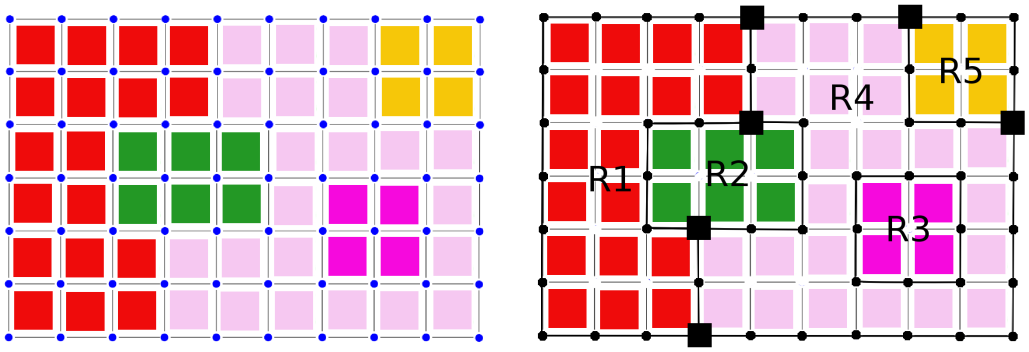
(a) 1 frontière commune (b) 2 frontières comunnes (c) inclusion

FIG. 2.8 – Différentes configurations caractérisées par une relation d'adjacence entre les régions bleues et marrons. Ces deux régions présentent une (a) ou 2 (b) frontières communes ou sont encore incluses (c) l'une dans l'autre.

Définition 32 Composante connexe

Une composante connexe d'une partition est définie comme un ensemble de régions adjacentes lui même inclu dans une région de la partition.

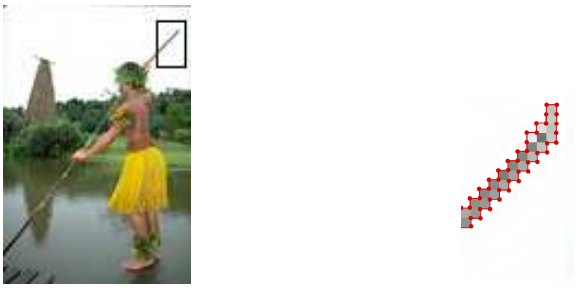
La Figure 2.9(b) présente l'ensemble des frontières interpixel de l'image Figure 2.9(a). Nous pouvons remarquer que la frontière entre les régions R_1 et R_2 est composée d'une courbe fermée, alors que la frontière entre les régions R_1 et R_4 est composée de deux courbes distinctes. Notons que les régions R_1 et R_3 n'étant pas adjacentes leur frontière est vide.



(a) Eléments interpixel (b) nœuds et segments

FIG. 2.9 – Partition représentée par des éléments interpixel (a) ainsi que la représentation de ses frontières interpixel avec des nœuds (carrés noirs) et des segments (lignes noires) .

Cette façon de définir les frontières, grâce à des entités sous-pixelaires va nous permettre de définir des régions, dans une image, au pixel près, comme illustré dans la Figure 2.10



(a) image naturelle d'un pêcheur

(b) zoom sur l'image (a)

FIG. 2.10 – Frontières interpixels obtenues sur une image naturelle .

2.3 Graphes et partitions

Nous présentons dans cette section quelques-uns des principaux modèles permettant la représentation d'images 2D segmentées en régions. Dans un premier temps (Section 2.3.1) nous présentons le Graphe d'Adjacence de Régions noté RAG (pour Region Adjacency Graph). Ce tout premier modèle de représentation d'images 2D segmentées en régions, a été une source d'inspiration pour de nombreux modèles. Nous poursuivrons ensuite (Section 2.3.2) en présentant la structure de graphes duaux, qui peut être considérée comme une évolution des RAG, apportant des solutions à certaines limitations de ces derniers. Nous terminerons cette étude des différents modèles à la Section 2.3.3, ou nous présenterons la structure de carte combinatoire et ses liens avec les frontières interpixel.

2.3.1 Graphe d'adjacence de régions

Le **Graphe d'adjacence de Régions** noté RAG, est une des premières structures utilisées pour coder des partitions et ainsi représenter des images 2D segmentées en régions, en rendant compte des relations d'adjacences entre les différentes régions de l'image.

(a) Image segmentée en régions.

(b) Rag correspondant

FIG. 2.11 – Image en 2D segmentée en régions (a) et le RAG correspondant (b)

L'un de ses principaux avantages tient au fait qu'il se définit très simplement. En effet, c'est un graphe simple (Def. 7), où chaque nœud représente une région de l'image segmentée et deux sommets sont reliés par une arête si les deux régions correspondantes sont adjacentes (Déf 31). Dans le cadre de méthodes ascendantes, la fusion de régions est codée par des opérations de contraction (Def. 12) et suppression (Def. 11, voir également section 2.4.3.a). Le codage de partitions par graphe d'adjacence est illustré par la Figure 2.11 présentant une image segmentée en régions avec le RAG correspondant.

Le RAG se positionne comme un moyen efficace de représenter et de coder les relations d'adjacences entre les régions. Il a l'avantage d'être simple à définir et de pouvoir s'étendre en dimension quelconque. Il représente également un moyen efficace pour mettre à jour les différents paramètres de régions lors de fusions successives.

Il possède néanmoins certaines limitations qui ne permettent pas de différencier certaines configurations ou relations.

– Il ne représente pas l'adjacence multiple. Sur notre exemple de la Figure 2.11, la région R_1 est adjacente à la région R_4 le long de deux courbes frontières (Déf. 30). Cette information n'est pas représentée dans le RAG

– Il ne permet pas de différencier les relations d'adjacence et les relations d'inclusion. Ceci est également illustré sur le RAG de la Figure 2.11(b) qui ne permet pas de différencier la relation d'adjacence entre les régions R_4 et R_5 et la relation d'inclusion existant entre R_3 et R_4.

Cette structure n'est donc pas à même de représenter certaines configurations rencontrées dans les images. Ceci est illustré par la Figure 2.12), sur laquelle on peut voir, que les deux panneaux, ayant pourtant une topologie différente, sont représentés par le même RAG.

(a) (b) (c)

FIG. 2.12 – Deux panneaux signalétiques (a)(b) avec le RAG correspondant à leur segmentation en régions (c)

Ces nombreuses limitations ont motivé l'étude d'une évolution du RAG, les graphes duaux.

2.3.2 Graphes duaux

Les graphes duaux ont été introduits, dans le cadre du codage de partition, principalement pour pallier aux limitations des RAG. L'idée principale est de conserver deux graphes en parallèle, une extension du RAG d'un côté, et son graphe dual de l'autre.

Le premier graphe est une simple extension du RAG, auquel nous allons ajouter des arêtes multiples et des boucles afin de coder les multiples frontières entre deux régions et caractériser les relations d'inclusion.

Plus précisément, dans le cadre du codage de partition d'image $P = \{R_1 \dots R_2\}$ d'un domaine de $\mathbb{Z}^2$, le dual de G noté $\overline{G}$ sera défini comme le graphe des frontières des régions connexes, tel que :

– chaque arête de $\overline{G}$ modélise une courbe-frontière (Def 30) entre deux régions adjacentes. Chacune de ces arêtes sera coupée par une arête de G représentant l'adjacence entre deux régions.
– chaque sommet de $\overline{G}$ modélise un point de jonction entre au moins trois courbes-frontière (ou avec le bord du domaine).
– chaque face de $\overline{G}$ représente une région de la partition P. Ces faces seront représentées dans G par un sommet.

Par rapport au graphe primal G correspondant au RAG étendu, le rôle des sommets et des faces est échangé dans son dual $\overline{G}$:

– les faces de $\overline{G}$ (correspondant aux régions), constituent les sommets du RAG

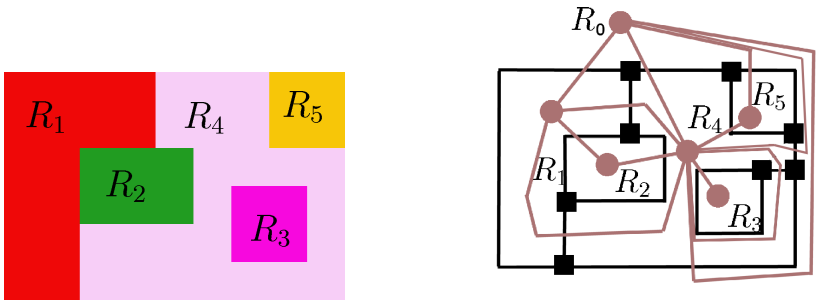
– les sommets de $\overline{G}$ (correspondant aux nœuds géométriques) constituent les faces du RAG.

Notons que l'aspect de dualité se retrouve également dans les opérations de contractions et suppressions d'arêtes. En effet une suppression d'arête dans G est équivalente à une contraction dans $\overline{G}$. Inversement une contraction dans G est équivalente à une suppression dans $\overline{G}$.

Ce type de structure admet également des boucles permettant de caractériser les relations d'inclusion. Ces boucles sont dûes au fait que le graphe primal ainsi que son dual doivent être tous deux connexes. De ce fait, une arête «fictive» reliant une région incluse à la région la contenant est ajouté dans le graphe dual. Cette arête devient une boucle autour de la région contenant l'autre, dans le graphe primal.

La Figure 2.13 présente un exemple de graphe dual de l'image de départ segmentée en régions. Nous avons représenté sur la Figure 2.13(b) le graphe dual en noir, alors que le graphe primal est représenté en marron. Nous pouvons observer sur cet exemple que le graphe primal est bien une extension du RAG, tenant désormais compte des adjacences multiples (il y a désormais deux arêtes entre les régions R_1 et R_4) et représentant les relations d'inclusion au moyen de boucles. La région R_4 contenant la région R_3 possède ainsi une boucle sur son sommet qui entoure le sommet R_3.

Notons également la présence, dans le graphe dual, d'une arête fictive reliant le bord de la région R_3 avec celui de la région R_5. Cette arête est fictive, du fait qu'elle ne représente pas, contrairement aux autres arêtes de ce graphe, une frontière entre deux régions. Par contre, cette arête est un pont de $\overline{G}$ et est par conséquent nécessaire pour conserver la connexité du graphe dual. Ce pont de $\overline{G}$ correspond à une boucle dans G. Cette boucle permet de caractériser le fait que la région correspondant au sommet incident à la boucle inclut au moins une région.



(a) Image segmentée en régions.

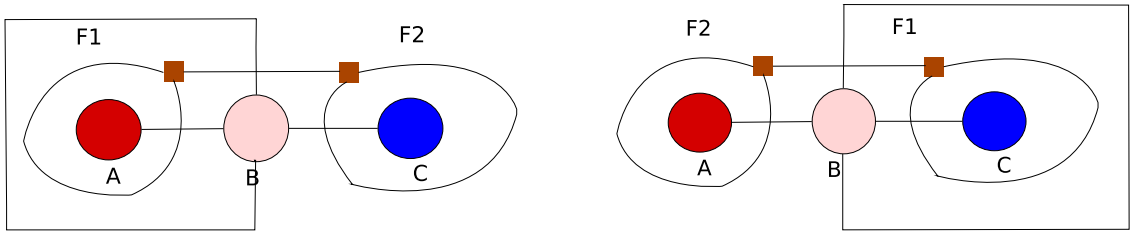
(b) Le graphe dual correspondant

FIG. 2.13 – Image 2D segmentée en régions (a) et le graphe dual correspondant (b)

Ce type de graphe possède néanmoins plusieurs inconvénients :

– Il est nécessaire de maintenir à tout moment deux graphes en parallèle , ce qui peut s'avérer coûteux en temps de calcul (il faut multiplier par deux toutes les opérations de mise à jour), mais également en espace mémoire (obligation de stocker deux structures).

– L'existence d'une boucle incidente à un sommet permet de caractériser le fait que la région associée à celui-ci inclue une région. En revanche l'existance d'une boucle ne permet pas de dire quel sommet va être inclu dans l'autre. Ceci est illustré dans la Figure 2.14 qui présente deux configurations définissant les mêmes relations d'adjacences entre les sommets et les faces. Il est donc impossible de décider si la boucle incidente à B entoure A ou C.



(a) (b)

FIG. 2.14 –
Deux graphes duaux illustrant l'impossibilité de caractériser les relations d'inclusion.

Ce type de structure a été largement utilisé pour modéliser de nombreux problèmes dans le domaine du traitement d'image [ENGLER00, KROPAT95A, KROPAT95C, KROPAT95B] et a également servi de base pour la définition de la strucuture de BRAG introduite par Guigues [GUIGUE03].

2.3.3 Les cartes combinatoires

Les cartes topologiques ont été étudiées dans les années 60 du point de vue combinatoire par Tutte (voir [TUTTE73, W.T.84] par exemple) et Walsh [T.72A, T.72B]).

Elles ont été introduites dans un premier temps pour répondre à des problèmes tels que l'énumération des familles de cartes en fonction de certains critères, par exemple : combien y a-t-il de cartes planaires à n sommets et m faces ? Cette recherche s'est heurtée rapidement à la complexité des calculs formels qu'elle nécessite (voir par exemple [TUTTE68]) et seules les cartes planaires ont conduit à des résultats explicites.

Les travaux de Cori dans les années 70 (voir par exemple [CORI75]) ont considérablement développé cette étude des cartes planaires. C'est la notion de carte combinatoire (introduite par Edmonds [EDMOND60]) qui est au départ de ces développements.

Nous présentons ici les définitions et les liens entre les notions de carte topologique et de carte combinatoire.

Les cartes combinatoires sont basées sur la définition des cartes topologiques qui codent la décomposition d'une surface en un ensemble fini de points $\mathcal{V}$ et un ensemble fini $\mathcal{E}$ de courbes ouvertes de Jordan. Nous baserons la suite de notre développement sur la définition des cartes topologiques donnée par Edmond [EDMOND60, GARETH78] au début des années 60.

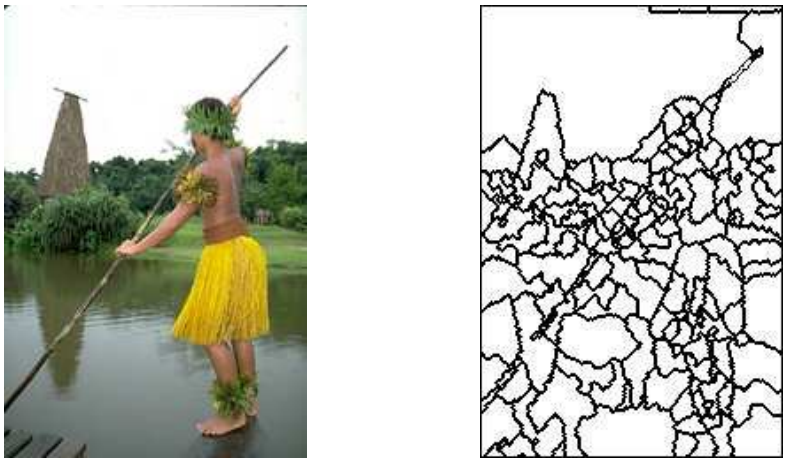
Définition 33 Carte topologique

Étant donné un ensemble non vide $\mathcal{E}$ d'espaces topologiques (appelés arêtes) chacun homéomorphe à l'intervalle $I = [0, 1]$ ou au cercle unité S^1, nous définissons l'ensemble $\mathcal{V} \subset \mathcal{G} = \cup_{e \in \mathcal{E}} e$ ($\mathcal{V}$ est l'ensemble des sommets) tel que si $\Delta e = e \cap \mathcal{V}$ et $e^{\#} = e \setminus \Delta e$ alors :

1. *Si e est homéomorphe à S^1 alors $|\Delta e| = 1$ (e est appelé une boucle) alors que si e est homéomorphe à $I = [0, 1]$ Δe contient une ou deux des extrémités de e (e est alors respectivement appelé une arête libre et un segment).*
2. *Pour toute arête distincte $e_1, e_2 \in \mathcal{E}$, $e_1^{\#} \cap e_2^{\#} = \emptyset$.*
3. *Pour tout $v \in \mathcal{V}$, le nombre d'arêtes e telles que $v \in \Delta e$ est fini.*

Pour illustrer cette définition, considérons la figure 2.15 et supposons que chaque contour de la figure 2.15(b) code une courbe réelle (le même type de raisonnement peut être conduit avec des courbes discrètes). L'ensemble $\mathcal{G}$ sera alors défini par l'union de l'ensemble des points de contours définis par ces courbes . Un sommet appartenant à $\mathcal{V}$ sera défini comme l'intersection de plusieurs contours. Une arête appartenant à $\mathcal{E}$ sera codée comme une courbe comprise entre deux sommets. La courbe est dite fermée si les deux sommets sont confondus,et est homéomorphe à l'ensemble S^1. C'est donc une boucle. Sinon la courbe est ouverte et est homéomorphe à l'intervalle $I = [0, 1]$, une telle courbe est appelée un segment.

Plus généralement, une carte topologique correspond au tracé sur une surface d'un ensemble de segments (éventuellement fermés) respectant les trois conditions suivantes :
- ils ont au plus deux extrêmités (condition 1)
- il ne peuvent se croiser que sur une extrêmité (condition 2)
- un nombre fini de segments se rencontrent en chaque sommet (condition 3)



(a) pecheur (b) Contours d'une segmentation

FIG. 2.15 – Ensemble des contours d'une segmentation illustrant la notion de carte topologique.

Cette définition étant un peu trop générale pour les applications qui vont suivre, nous supprimons la possibilité des arêtes libres.

Chaque arête comprend donc deux extrémités appelées brins. Chaque brin appartient par construction à une seule arête. De plus, comme chaque sommet est situé aux extrémités

d'une arête (condition 2), un brin n'appartient qu'à un seul sommet alors qu'un sommet peut correspondre à plusieurs extrémités d'arêtes et donc à plusieurs brins. Le nombre de brins d'un sommet est appelé son degré. Le degré de tout point p intérieur à un segment ($p \in e^{\#}, e \in \mathcal{E}$) est fixé à 2 par convention.

On définit sur l'ensemble des brins une application α qui associe à chaque brin d'une arête le brin restant dans la même arête. L'application α est donc une involution permettant de passer d'une extrémité d'une arête à l'autre (figure 2.16(a)) et ses cycles codent les arêtes du graphe.

Pour définir les sommets nous supposerons que :

Propriété 1 *Pour tout point $p \in \mathcal{G}$, de degré k, il existe un voisinage V_p de p dans $\mathcal{S}$ et un homéomorphisme ψ_p de V_p dans $D = \{z \in \mathbb{C} \mid |z| < 1\}$ tel que $\psi_p(p) = 0$ et $\psi_p(V_p \cap \mathcal{G}) = \{z \in \mathbb{C} \mid z^k \in [0,1[\subset \mathbb{R}\}$.*

Cette propriété nous permet de définir un ordre cyclique sur l'ensemble des brins de p en assimilant l'ensemble $V_p \cap \mathcal{G}$, défini pour tout point p, à un ensemble de rayons répartis autour de ce point. Cet ordre définit une permutation sur l'ensemble des brins d'un sommet. Chaque brin appartenant à un seul sommet, la composition des cycles, définis sur chacun des sommets, définit une permutation σ sur l'ensemble des brins. On peut voir cette permutation σ comme une rotation permettant de passer d'un brin, au brin suivant, en tournant dans le sens positif autour d'un sommet (figure 2.16(b)).

(a) α (b) σ

FIG. 2.16 – Illustration des permutations α et σ

Lemme 1 *Étant donnée une carte topologique connexe $(\mathcal{V}, \mathcal{E})$ et deux permutations α et σ sur l'ensemble des brins telles que deux brins b et b' appartiennent au même cycle*
- *de α s'ils appartiennent à une même arête et*
- *de σ s'ils appartiennent à un même sommet,*

il existe une bijection naturelle entre les arêtes de $\mathcal{E}$ et les cycles de la permutation α ainsi qu'entre les sommets de $\mathcal{V}$ et les cycles de la permutations σ.

Preuve:

Voir [CORI75] □

Cori [CORI75] montre qu'à partir d'une carte topologique $(\mathcal{V}, \mathcal{E})$ connexe, on peut :
- définir un ensemble de brins $\mathcal{D}$.
- coder les arêtes $e \in \mathcal{E}$ par les cycles de la permutation α
- coder les sommets $v \in \mathcal{V}$ par les cycles de la permutation σ.

Le triplet défini par $(\mathcal{D}, \sigma, \alpha)$ est appelé une **carte combinatoire** [CORI75].

Définition 34 Carte combinatoire

Une carte combinatoire $G = (\mathcal{D}, \sigma, \alpha)$ est définie par un ensemble de brins $\mathcal{D}$ et deux permutations σ et α sur $\mathcal{D}$ telles que α et une involution sans point fixe.

$$\forall b \in \mathcal{D}\ \alpha^2(b) = b \text{ et } \alpha(b) \neq b.$$

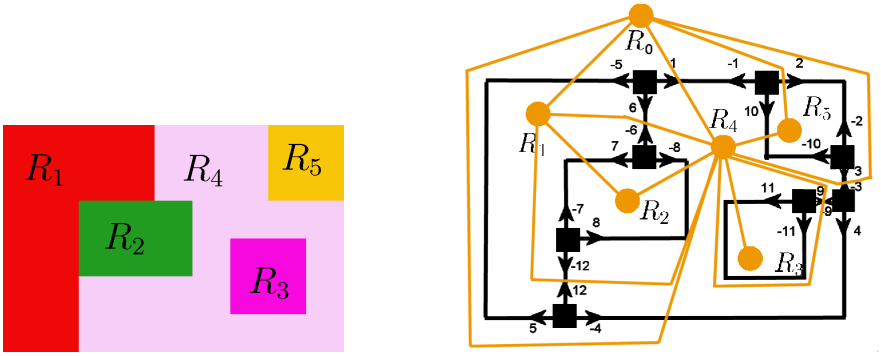
Un sommet possédant un brin b est codé par le cycle de σ contenant b. Ce cycle est noté $\sigma^*(b)$. De même, l'arête contenant un brin b est codée par le cycle de α contenant b. Ce cycle est noté $\alpha^*(b)$. Plus généralement, pour une permutation π, le π-cycle d'un brin b sera noté $\pi^*(b)$.

En termes de cartes, la notion d'ensemble d'arête est codée par des ensembles symétriques de brins :

Définition 35 Ensemble symétrique de brins

Soit $G = (\mathcal{D}, \sigma, \alpha)$ une carte combinatoire, un ensemble $\mathcal{B} \subset \mathcal{D}$ sera appelé un ensemble symétrique de brins de G si $\alpha^*(\mathcal{B}) = \mathcal{B}$.

Un codage d'une partition du plan par une carte combinatoire est présenté sur la figure 2.17 où l'involution α est implicitement codée par le signe. Ce type de codage implicite est souvent utilisé dans les applications [BR96].



(a) Image segmentée en régions.

(b) carte combinatoire

FIG. 2.17 – Image 2D (a) et la carte combinatoire correspondant à une segmentation en régions (b)

Notez que le codage d'une partition du plan par des graphes simples (section 2.3.1) ou des graphes duaux (section 2.3.2) ne fait pas appel à la notion de carte topologique. Cette correspondance explicite entre les arêtes de la carte combinatoire et celles de la carte topologique nous garantit *a priori* une description correcte de la partition.

Nous illustrons ce propos sur la figure 2.18 qui représente deux partitions différentes dans lesquelles seuls les sommets A et B ont été échangés. On peut noter d'une part que ces partitions ne seraient pas représentables avec des graphes simples du fait de la présence

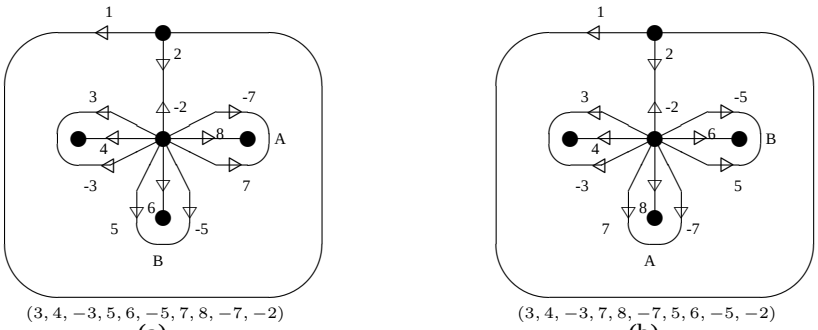


FIG. 2.18 – Deux cartes combinatoires distinctes non codables par des graphes simples et non distinguables par des graphes duaux. Le σ cycle du sommet central est indiqué sous chaque figure.

de boucles. D'autre part ces deux partitions ne diffèrant que sur l'orientation, elles seraient codées de façon semblable par des graphes duaux.

Comme nous venons de le voir une carte combinatoire peut être définie comme un codage d'une partition connexe du plan par une carte topologique. Elle peut également être définie comme un codage particulier d'un graphe planaire. Ceci va nous conduire à définir le dual d'une carte combinatoire :

Définition 36 Dual d'une carte combinatoire

Le dual d'une carte combinatoire $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ connexe noté $\overline{G}$ est une carte combinatoire définie par le triplet $\overline{G} = (\boldsymbol{\mathcal{D}}, \varphi = \sigma \circ \alpha, \alpha)$.

Les cycles de la permutation $\varphi = \sigma \circ \alpha$ codent les faces de la carte combinatoire. Gareth [GARETH78] montre que si G est connexe, il existe une bijection naturelle entre les cycles de φ et l'ensemble des composantes connexes de $\mathcal{S}/\mathcal{G}$.

On peut facilement démontrer, que comme pour tout graphe, $\overline{\overline{G}} = G$: α étant une involution, nous avons $\varphi \circ \alpha = \sigma \circ \alpha \circ \alpha = \sigma$. Donc :

$$\overline{\overline{G}} = (\boldsymbol{\mathcal{D}}, \varphi \circ \alpha, \alpha) = (\boldsymbol{\mathcal{D}}, \sigma, \alpha) = G.$$

La suppression d'une arête se traduit, en terme de cartes par une simple modification de la permutation σ [BRUN99] :

Définition 37 Suppression d'une arête

Soient une carte combinatoire $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ et un brin $b \in \boldsymbol{\mathcal{D}}$ qui ne définit ni une boucle directe ni un pont dans G. La carte issue de la suppression de l'arête $\alpha^*(b)$ est notée $G \setminus \alpha^*(b)$ et définie par le triplet $(\boldsymbol{\mathcal{D}} - \{b, \alpha(b)\}, \sigma', \alpha')$ avec :

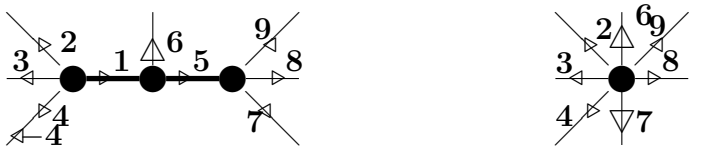
$$\left\{ \begin{array}{lcl} \forall b' \in \{\sigma^{-1}(b), \sigma^{-1}(\alpha(b))\} \sigma'(b') & = & \sigma'(b) \\ \sigma'(\sigma^{-1}(b)) & = & \sigma(b) \\ \sigma'(\sigma^{-1}(\alpha(b))) & = & \sigma(\alpha(b)) \end{array} \right.$$

Comme nous l'avons vu dans la Section 2.4.3.b la suppression d'un ensemble d'arêtes dans un graphe G se traduit par la compression des arêtes correspondantes dans son dual $\overline{G}$. Inversement la compression d'un ensemble d'arêtes dans G se répercute par la suppression des arêtes correspondantes dans $\overline{G}$. Du fait que dans le cadre des cartes combinatoires les arêtes d'une carte et son dual son définis à partir du même ensemble de brins (Déf. 36), on peut définir l'opération de contraction comme étant une opération de suppression effectuée dans le dual :

Définition 38 Contraction d'un brin

Soient une carte $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ et un brin $b \subset \boldsymbol{\mathcal{D}}$. La carte contractée est définie par :

$$G' = G/\boldsymbol{\mathcal{D}}' = \overline{\overline{G} \setminus b}.$$



(a) G (b) $G/\alpha^*(1,5)$

FIG. 2.19 – Opération de contraction. Les arêtes correspondant à des brins contractés sont représentées en traits épais

La Figure 2.19 illustre une opération de contraction. On peut noter que tout comme une opération de suppression, la contraction préserve l'orientation des brins autour du sommet contracté. Cette propriété permet, si les sommets de la carte combinatoire codent les régions d'une partition, d'obtenir les régions que l'on rencontre en tournant dans le sens anti-horaire en utilisant l'ordre cyclique défini sur chaque sommet par la permutation σ.

Nous pouvons également retranscrire quelques notions usuelles de la théorie de graphe en terme de carte combinatoire. Un brin $b \in \boldsymbol{\mathcal{D}}$, dans les configurations suivantes sera appelé :

- une boucle si $\alpha(b) \in \sigma^*(b)$;
- un pont si $\alpha(b) \in \varphi^*(b)$;

D'autres configurations de brin propres aux graphes planaires peuvent également être caractérisées en termes de brins dans les cartes combinatoires

- une boucle vide si $\sigma(b) = \alpha(b)$;
  En effet, on a alors $\varphi(\alpha(b)) = \sigma(b) = \alpha(b)$. Le φ-cycle de $\alpha(b)$ est donc réduit à $(\alpha(b))$. Le brin $\alpha(b)$ code donc bien un sommet de la carte duale de degré 1 ce qui est compatible avec la définition 17 (voir également le brin -1 Fig. 2.3.3(a) et (b)).
- un brin pendant si $\sigma(b) = b$.
  Notez que si $\sigma(b) = b$ alors $\varphi(\alpha(b)) = \sigma(b) = b$. Un brin pendant correspond donc à une boucle vide dans le dual (et vice versa, voir le brin -4 Fig. 2.3.3(a) et (b)).

(a) : $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ (b) : $\overline{G} = (\boldsymbol{\mathcal{D}}, \varphi, \alpha)$

FIG. 2.20 – Les arêtes $\alpha^*(1)$,$\alpha^*(2)$,$\alpha^*(3)$,$\alpha^*(4)$ codent respectivement une boucle directe, un pont, une boucle et une arête pendante de la carte combinatoire G(a). Chacune de ces arêtes devient une autre configuration particulière dans la carte duale $\overline{G}$(b)

– une arête double si $\varphi^2(b) = b$.
  Le φ-cycle de b est donc égal à $(b, \varphi(b))$ et b code bien un sommet dual de degré 2. Une telle caractérisation est compatible avec la définition 18. Notez que $\varphi^2(\varphi(b)) = \varphi(\varphi^2(b)) = \varphi(b)$. Donc, si b est une arête double, $\varphi(b)$ l'est également.

Par extension, nous dirons qu'une arête est une boucle, une boucle directe, un pont ou une arête pendante si un des ses brins vérifie la propriété correspondante.

Chacune de ces caractérisations d'arête dans la carte initiale correspond à un autre type de configuration particulière dans la carte duale. Ces correspondances sont explicitées dans la tableau 2.1 (voir également Fig. 2.3.3) [BRUN99].

Configuration dans G	Configuration dans $\overline{G}$
boucle	pont
pont	boucle
boucle directe	arête pendante
arête pendante	boucle directe

TAB. 2.1 – Relations entre les configurations particulières d'arêtes dans la carte initiale et son dual

De même que pour les graphes duaux, les boucles permettent aux cartes combinatoires de caractériser l'existence de relations d'inclusion. En utilisant le plongement des arêtes ainsi que l'orientation du plan, explicitement encodé, il est possible d'obtenir l'ensemble des régions incluses dans une région donnée [BRUN05A] et ainsi lever l'ambiguité rencontrée avec les graphes duaux dans certaines configurations.

Tout comme les graphes duaux, les cartes combinatoires permettent de coder les adjacences multiples et de caractériser les relations d'inclusions.Toutefois, les cartes présentent des avantages supplémentaires permettant également de résoudre les problèmes rencontrés par les graphes duaux. Du fait que les cartes combinatoires admettent un codage implicite du dual (Déf.36) , elles ont l'avantage de conserver l'ensemble des informations fournies par les graphes duaux en maintenant une seule structure, ce qui évite les doubles mises à jour. Enfin cette structure permet de caractériser complètement les relations d'inclusion [BRUN06] et de profiter de l'orientation du plan pour différencier certaines configurations, indissociables avec des graphes duaux (Figure.2.18).

2.4 Codage de Hiérarchies

2.4.1 Relations hiérarchiques

Le terme hiérarchie, tiré du grec *hieros* signifiant «sacré» et *arkho* signifiant «règle» est défini comme un système de classement et d'organisation de choses ou de personnes. Chaque élément de ce système (excepté l'élément le plus haut dans cette hiérarchie) est subordonné à un seul autre élément.

Il peut être utile pour la suite de ce développement de définir la notion d'ordre. Un ordre est une relation binaire permettant de comparer les éléments d'un ensemble entre eux. On peut par exemple citer l'ordre naturel défini sur les nombres réels $\leq$ ou encore l'inclusion ensembliste $\subseteq$.

Définition 39 Ordre

Une relation binaire $\preceq$ sur un ensemble X est un ordre si :

$$\forall x_i, x_j \in X \quad (x_i \preceq x_j) \wedge (x_j \preceq x_i) \Leftrightarrow (x_i = x_j) \tag{2.4}$$

$$\forall x_i, x_j, x_k \in X \quad (x_i \preceq x_j) \wedge (x_j \preceq x_k) : (x_i \preceq x_k) \tag{2.5}$$

de plus l'ordre $\preceq$ est total si :

$$\forall x_i, x_j \in X (x_i \preceq x_j) \vee (x_j \preceq x_i) \tag{2.6}$$

Dans un ensemble totalement ordonné, tout couple d'élements appartenant à cet ensemble peut être comparé (Equation 2.6).

Définition 40 Relation de couverture

Soit $(X, \preceq)$ un ensemble ordonné et $x_i, x_j \in X$. On dit que x_i est couvert par x_j (ou x_j couvre x_i) et on note $x_i \prec x_j$ si $x_i < x_j$ et si pour tout x tel que $x_i \leq x < x_j$ on a $x = x_i$. Autrement dit, il n'existe pas $x \in X$ tel que $x_i < x < x_j$.

Définition 41 Ordre hiérarchique

Une relation d'ordre $\preceq$ sur un ensemble X est un ordre hiérarchique si :

$$\forall x_i, x_j, x_k \in X \quad (x_i \preceq x_j) \wedge (x_i \preceq x_k) \Leftrightarrow (x_j \preceq x_k) \vee (x_k \preceq x_j) \tag{2.7}$$

*Le couple $(X, \preceq)$ est appelé une **forêt**.*

L'ensemble des prédécesseurs d'un élément, sur une forêt, est donc totalement ordonné (Définition 41).

Nous traiterons, dans la suite de ce manuscrit, des hiérarchies dont les éléments sont des régions avec pour relation d'ordre l'inclusion. Nous nous intéresserons donc, dans la suite de cette section, aux hiérarchies, représentables par des arbres, dont la relation d'ordre est déterminée par la relation d'inclusion entre les parties d'un ensemble.

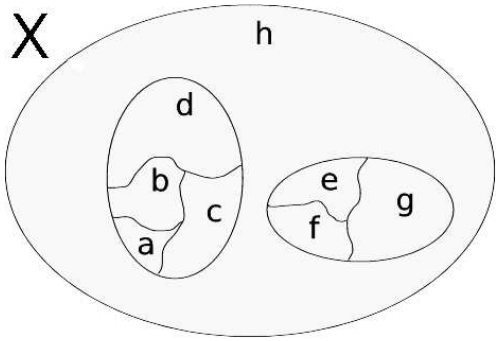


FIG. 2.21 – Ensemble de parties hiérarchisées

Deux parties x et x' d'un ensemble X sont dites hiérarchisées si et seulement si elles sont soit emboitées, soit disjointes (i.e $x \cap x' \in \{x, x', \emptyset\}$). Cette notion peut être généralisée à un ensemble H de parties, qui sera dit **hiérarchisé** si ses éléments le sont tous deux à deux.

Définition 42 Hiérarchie

Soit X un ensemble fini. Un ensemble H hiérarchisé de parties de X sera appelé une hiérarchie s'il vérifie :

(i) $\emptyset \notin H$.

(ii) $X \in H$.

(iii) $\forall x \in X \quad : \quad \{x\} \in H$.

*X est appelé le **sommet** de la hiérarchie et l'ensemble $B(H) = \{\{x\}\}_{x \in X}$ est appelé la **base** de la hiérarchie.*

Notons que l'ensemble $B(H)$ correspond à la sur-segmentation absolue de X et que les élements de $B(H)$ sont appelés les **atomes**. Les parties de H qui ne sont ni le sommet, ni les atomes sont dites **intérieures** à H.

Définition 43 Dendrogramme

Un dendrogramme est un graphe planaire non orienté. Les classes sont représentées par des points, les éléments terminaux se trouvent en bas, le sommet en haut et les traits indiquent l'ordre hiérarchique entre les classes. (voir Figure 2.22)

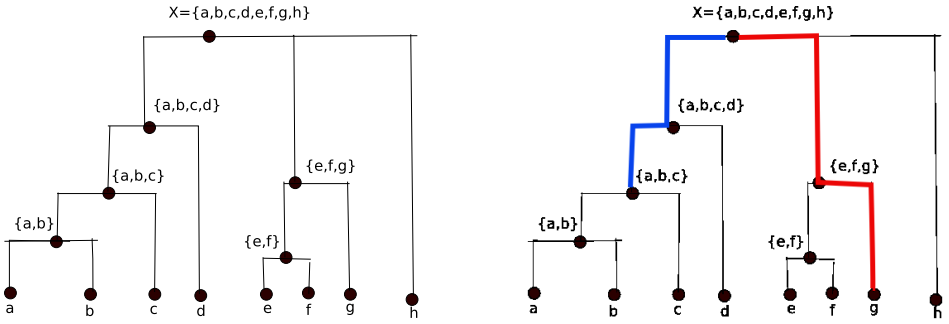
Une hiérarchie H, peut être représentée par une structure arborescente telle que le dendrogramme (voir Figure 2.22 et Déf. 43). Nous parlerons alors des nœuds et des arêtes d'une hiérarchie, les atomes sont alors appelés *feuilles* et le sommet *racine*.

Les feuilles correspondent à l'ensemble des éléments minimaux d'un arbre $(X, \preceq)$ défini par :

$$\{x \in X \quad | \quad \forall x' \in X \quad : \quad x' \preceq x : (x' = x)\} \tag{2.8}$$

alors que le sommet correspond à l'élément maximal de $(X, \preceq)$ défini comme :

$$\{x \in X \quad | \quad \forall x' \in X \quad : \quad x \preceq x' : (x' = x)\} \tag{2.9}$$



(a) dendrogramme (b) branches

FIG. 2.22 – (a) Représentation d'une hiérarchie sous forme d'un dendrogramme.(b) La branche de g est représentée en rouge, celle de $\{a, b, c\}$ en bleu.

Tout élément x d'une hiérarchie finie, ormis le sommet, possède un élément unique couvrant, généralement appelé le père de x, noté $p(x)$. Inversement, tout élément x qui n'est pas un atome, couvre un certain nombre d'éléments de H, appelés les fils de x, noté $F(x)$. Le nombre de fils d'un sommet, $|F(x)|$, s'appelle l'**arité** et une hiérarchie dont tous les sommets sont de même arité peut être représentée par un arbre n-aire. On appelle **branche** de x (voir (Figure 2.22(b)), la chaîne reliant x au sommet de la hiérarchie. On peut définir le **niveau** d'un élément d'une hiérarchie, comme étant la longueur du plus long chemin, le reliant à la base. Par extension, nous définirons le nombre de niveaux d'une hiérarchie comme étant le niveau de son sommet.

Définition 44 Pré-hiérarchie

*Une **pré-hierarchie** sur X est un ensemble de parties hiérarchisées de X qui contient $\{x\}$ pour tout $x \in X$, mais pas nécessairement lui-même.*

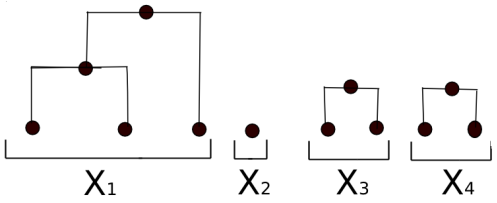


FIG. 2.23 – Une pré-hiérarchie

Si nous nous intéressons à présent à la notion de partition (Def. 29), il peut être utile de définir une relation de **finesse**, que l'on notera $\leq$ définie à partir de l'ordre sur les parties d'une partition telle que

$$\forall (A, B) \in \mathbb{P} \quad | \quad A \leq B \quad \Leftrightarrow \quad \forall x \in A \quad \exists x' \in B \quad | \quad x \subseteq x' \tag{2.10}$$

où $\mathbb{P}$ désigne l'ensemble des partitions de l'image.

Nous disons alors, quand $A \leq B$, que la partition A est plus fine que B ou que A est une sur-partition de B. Inversement B est une sous-partition de A et est moins fine que A.

La relation de finesse $A \leq B$ peut également exprimer que A peut être obtenue à partir de la partition B en découpant certaines parties, ou qu'inversement B peut être obtenue en réunissant certaines parties de A. A partir de cette relation de finesse nous pouvons définir la notion de *partitions emboitées*.

Définition 45 Partitions emboitées

- *Nous dirons de deux partitions A et B qui sont ordonnées pour la relation de finesse $\leq$ (équation 2.10), qu'elles sont emboitées.*
- *Une ensemble de partitions $\mathbb{P} = \{P_1, ..., P_k\}$ tel que*

$$\forall i \in \{1, \ldots, k-1\} \quad P_i < P_{i+1}$$

est appelé une séquence de partitions emboitées.

Une hiérarchie peut représenter une séquence de partitions monotones $\mathbb{P} = \{P_1, ..., P_k\}$ d'un ensemble X telle que $P_1 \leq \cdots \leq P_k$. De façon classique, P_1 est égal au sur-partitionnement absolu $B(H)$ de X alors que P_k correspond à la sous-partition absolue de X correspondant au sommet de la hiérarchie. Dans ce cas on dit que $\{P_i\}_{i=1,...,k}$ est une *séquence complète*. Notons cependant qu'une hiérarchie n'est pas équivalente à une séquence de partitions. En effet, plusieurs séquences peuvent fournir une même hiérarchie. En traitement d'image, l'amalgame entre pyramide de segmentation et hiérarchie de régions est souvent fait. Notons par contre que si l'on dispose d'une hiérarchie H sur un ensemble X , il est possible d'obtenir une partition de X en effectuant une coupe dans H.

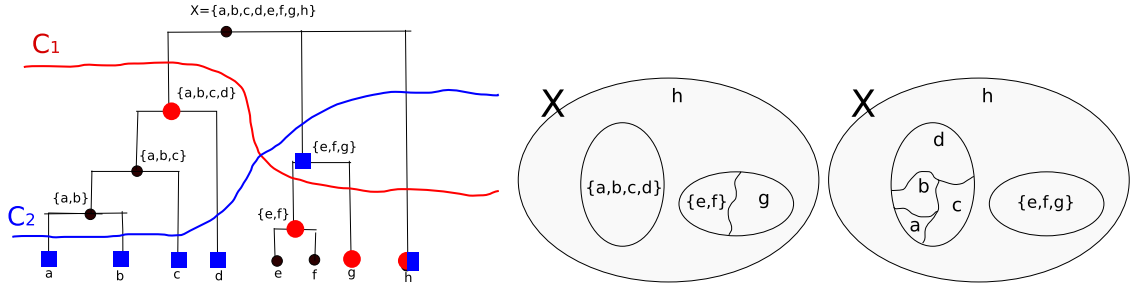
Définition 46 Coupe dans une hiérarchie

Soit H une hiérarchie sur un ensemble X. Une coupe dans H définit une partition de X dans laquelle chaque élément de la partition appartient à H

Nous noterons $\mathbb{C}ut(H)$ l'ensemble des coupes de H. $\mathbb{C}ut(H)$ correspond à l'ensemble des partitions de X que l'on peut construire avec des parties de H.

Si l'on interprète cette définition d'un point de vue graphique, une coupe dans une hiérarchie H correspond à un ensemble de nœuds de H, avec pour contrainte que la coupe doit intersecter une et une seule fois chaque branche de H(voir Figure 2.24). En effet comme les nœuds le long d'une branche sont ordonnés suivant la relation d'inclusion, le fait d'intersecter plusieurs fois une branche revient à prendre plusieurs nœuds dans cette branche. Le résultat de la coupe ne fournit plus une partition de l'ensemble de départ, car au moins une région est incluse dans une autre. Inversement le fait de ne pas intersecter une branche revient à suprimer une partie de l'ensemble de départ et ne fournit pas non plus une partition.

Une hiérarchie admet en général un très grand nombre de coupes. Deux coupes ne sont généralement pas ordonnées par finesse (équation 2.10), c'est le cas par exemple des coupes présentées sur la Figure 2.24. Par contre elle peuvent entretenir une relation d'*ordre local*. C'est également le cas des coupes présentées sur la Figure 2.24 où sur certaines parties du domaine, la coupe C_1 est une sur-partition de C_2 alors qu'au contraire sur d'autres parties, C_1 est une sous-partition de C_2.



(a) hiérarchie H (b) coupe C_1 (c) coupe C_2

FIG. 2.24 – (a) Coupes dans une hiérarchie. Les partitions correspondantes sont représentées sur (b) pour la coupe C_1 et en (c) pour la coupe C_2

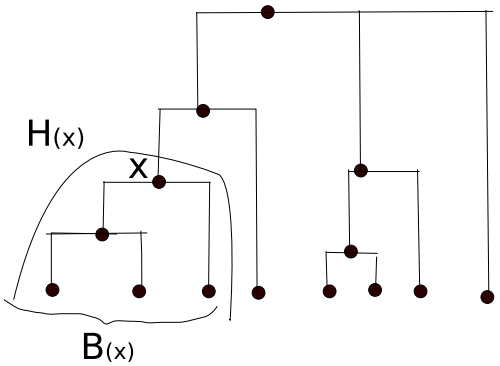


FIG. 2.25 – Hiérarchie partielle. $H(x)$ est la hiérarchie partielle issue de x avec $B(x)$ comme base.

Définition 47 Hiérarchie partielle

Soit H une hiérarchie sur un ensemble X. Pour tout $x \in H$, $H(x)$ désignera une ***hiérarchie partielle*** *issue de x*

$$H(x) \quad = \{x' \in H | x' \subseteq x\}$$

et $B(x)$ la ***base partielle*** *issue de x*

$$B(x) \quad = \{x' \in B(H) | x' \subseteq x\}$$

La Figure 2.25 illustre cette définition.

Pour poursuivre ce développement, l'étude d'un type de distance, appelé ultramétrique [ARBELA05] peut s'avérer utile. Un **espace ultramétrique** est un espace pseudo-métrique ou l'inégalité triangulaire est remplacée par la plus forte **inégalité ultramétrique** :

$$\forall (a, b, c) \in \mathbb{X} \quad \delta(a, b) \leq max\{\delta(a, c), \delta(c, b)\} \tag{2.11}$$

D'un point de vue géométrique, la relation précédente indique que tous les points d'une sphère sont équidistants, deux sphères qui s'intersectent sont nécessairement égales et que tous les triangles sont soit isocèles, soit équilatéraux.

Définition 48 Hiérarchie indicée

Une hiérarchie indicée est définie comme la donnée d'un couple $I = (H, \lambda)$, où H est une hiérarchie et λ est un critère croissant sur H et nul sur ses feuilles, appelé son indice.

La figure 2.26 illustre les différentes étapes de construction d'une hiérarchie indicée, d'un ensemble dont les éléments sont dans ce cas les points A, B, C, D, E, F, G. Ce type de hiérarchie, utilisant comme critère de regroupement la distance minimale entre les classes, est appelé *hiérarchie du saut minimum* [PAPKA99].

Benzécri a prouvé dans [BENZEC73] qu'il existe une bijection entre la classe des hiérarchies indicées et celle des espaces ultramétriques. Si (H, λ) est une hiérarchie indicée, sur un ensemble X, la fonction $\lambda : X^2 \rightarrow \mathbb{R}^+$ définie comme

$$\delta(x, x') = max(\lambda(x), \lambda(x')) \tag{2.12}$$

est une distance ultramétrique sur l'ensemble X. H est alors la hiérarchie induite par λ. La notion de hiérarchie indicée et d'espace ultramétrique est donc bien équivalente.

Il existe une infinité d'ultramétries associées à une hiérarchie H. Nous pouvons cependant noter deux ultramétries triviales. L'ultramétrie correspondant aux $niveaux$ des ensembles de H ainsi que l'ultramétrie correspondant aux cardinaux des ensembles. Ces deux ultramétries sont de toute évidence des critères croissants. Notons également que lorsque nous dessinons une hiérarchie sous forme d'arbre, nous utilisons, souvent machinalement, un critère croissant sur H, qui nous permet d'affecter aux éléments de H une

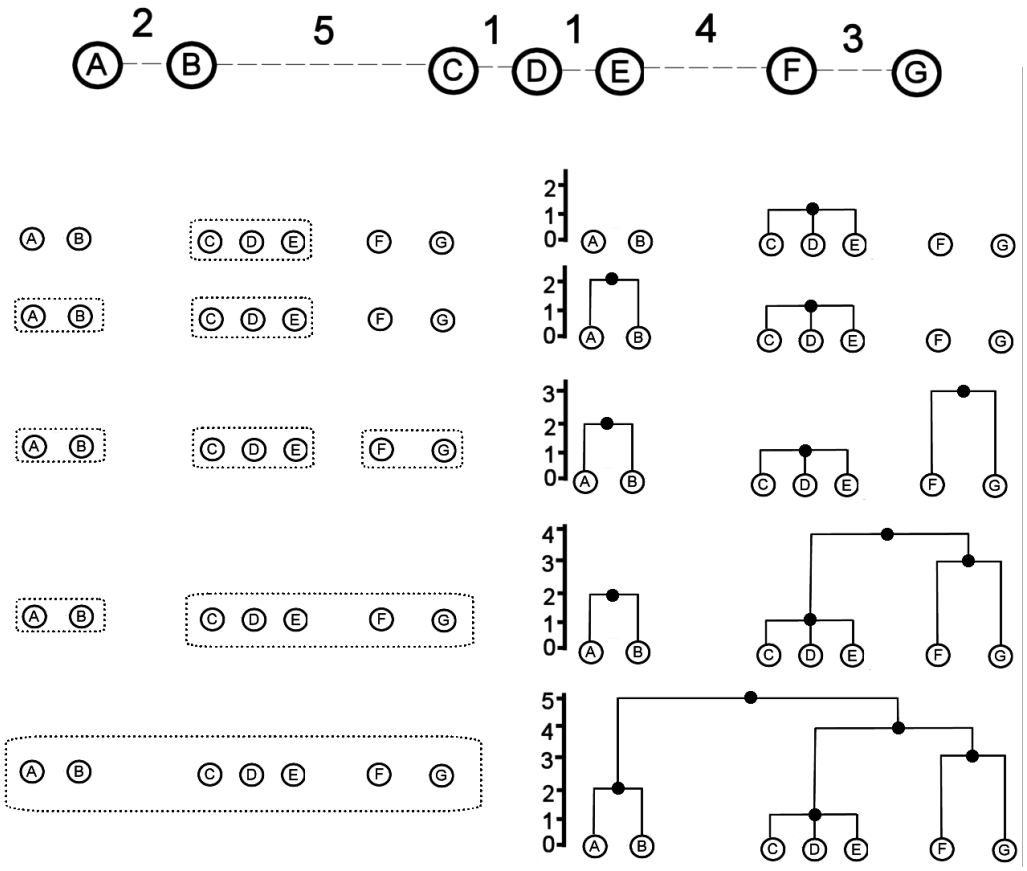


FIG. 2.26 – Exemple de construction d'une hiérarchie indicée.

ordonnée correspondant à leur niveau. L'utilisation inconsciente d'un critère croissant permet de placer les pères au dessus de leurs fils. Le critère du niveau est le plus utilisé car il permet d'obtenir la représentation la plus tassée verticalement.

Si maintenant nous nous restreignons aux hiérarchies munies d'un indice λ, ce type de hiérarchie admet une représentation naturelle consistant à affecter à chaque nœud $x \in H$ une ordonnée correspondant à son indice $\lambda(x)$. Ce type de représentation nous permet de définir la notion de coupe naturelle dans une hiérarchie indicée.

Définition 49 *Coupe naturelle*

Soit H une hiérarchie indicée munie d'un indice λ sur un ensemble X. Nous pouvons associer à (H, λ) une famille de coupes remarquables appelée famille des ***coupes naturelles****.*

La coupe naturelle C_d de hauteur $d \geq 0$ de (H, λ) est définie comme l'ensemble des parties de H d'indice inférieur à d et maximales pour la relation d'inclusion :

$$C_d = \{x \in H \quad | \quad \lambda(x) \leq d \text{ et } \nexists x' \in H \text{ tel que } x \subset x' \text{ et } \lambda(x') \leq d\}$$

Si l'on interprète ce résultat graphiquement, la coupe naturelle C_d correspond à la coupe graphique de la représentation naturelle selon la droite horizontale d'ordonnée d. La Figure 2.27 illustre cette notion de coupe naturelle dans une hiérarchie indicée. Les traits en pointillés matérialisent des coupes naturelles et les partitions correspondant à chacune d'elles sont représentées tout autour. La façon de construire de telles hiérarchies de régions étant une des contributions de ce mémoire, nous y reviendrons plus en détail dans la Section 2.6 ainsi que dans le Chapitre 3.

Intéressons-nous à présent à une famille de partitions d'une image I : $A(I) = (P_\lambda)_{\lambda \in \mathbb{R}^+}$ indicée par un paramètre d'échelle λ. On dira que cette famille de partitions correspond à une **structure causale** [GUIGUE06] si elle vérifie les conditions suivantes :

Définition 50 Structure Causale

On dira qu'une famille de partitions $(P_\lambda)_{\lambda \in \mathbb{R}^+}$ d'une image I définit une structure causale ssi pour tout couple $(\lambda_1, \lambda_2) \in \mathbb{R}^{+2}$ tel que $\lambda_2 \geq \lambda_1$, il existe un difféomorphisme ϕ de I tel que $\phi(\partial P_{\lambda_2}) \subset \partial P_{\lambda_1}$. Le symbole ∂P_λ désignant les frontières de la partition P_λ.

Autrement dit, si $\lambda_2 \geq \lambda_1$, les frontières de P_{λ_2} sont incluses modulo un morphing (codé par ϕ) dans celles de P_{λ_1}. Une telle famille de partitions vérifie le **principe de causalité** défini par Witkin [WITKIN84]. Ce principe stipule qu'une diminution de précision d'analyse (augmentation de l'échelle) ne peut créer d'information, donc toutes les structures présentes à une échelle grossière doivent trouver une «cause» dans les échelles plus fines.

Si nous imposons un principe plus fort à savoir l'interdiction de délocaliser les frontières ($\phi = Id$, l'identité de I), nous pouvons définir la notion **d'analyse multi-échelle non biaisée** [GUIGUE06]. Cette notion correspond au *principe de causalité* tel qu'il a été énoncé par Koepfler [KOEPFL94] :

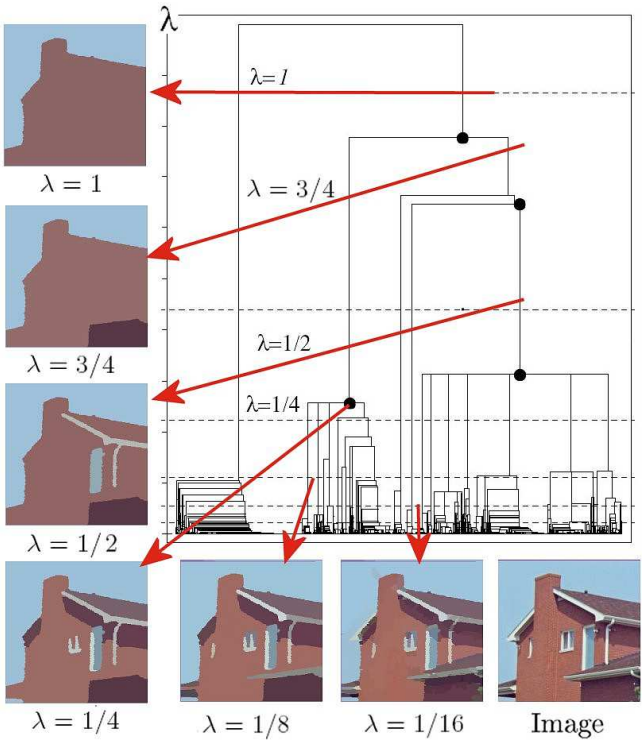


FIG. 2.27 – Coupes naturelles dans une hiérarchie de régions indicée par un facteur d'échelle positif.

Définition 51 Analyse multi-échelle non biaisée

Une famille de partitions $A(I) = (P_\lambda)_{\lambda \in \mathbb{R}^+}$ *d'un domaine* D *telle que*

$$\forall (\lambda, \lambda') \in \mathbb{R}^{+2} \quad \lambda' \geq \lambda \quad : \quad P_{\lambda'} \geq P_\lambda \tag{2.13}$$

sera appelé une analyse multi-échelles non biaisée.

Un algorithme A qui, à toute image I définie sur un domaine D, associe une analyse multi-échelle non biaisée sera simplement appelé un algorithme de segmentation multi-échelles. Ce type d'analyse peut être représenté *exactement* par les coupes naturelles d'une hiérarchie indicée. En effet, en considérant un algorithme de segmentation multi-échelles A ainsi qu'une image I définie sur un domaine D borné, avec $|D| = n$. Comme D possède n éléments, A renverra au maximum n partitions distinctes

$$\{P_1 < P_2 < \cdots < P_k\} \quad \text{avec} \quad k \leq n$$

Si nous considérons maintenant pour chaque partition P_i l'ensemble des valeurs χ de λ pour lesquelles A produit la partition P_i :

$$\chi_i = A^{-1}(P_i) \tag{2.14}$$

A étant défini sur $\mathbb{R}^+$, les χ_i constituent une partition de $\mathbb{R}^+$. Si en plus le principe de causalité est respecté on obtient :

$$\forall (i,j) \in [0,\ldots,k]^2 \quad \forall (x,x') \in \chi_i \times \chi_j : \quad i < j : x < x'. \tag{2.15}$$

Les χ_i constituent donc une suite d'intervalles ordonnés formant une partition de $\mathbb{R}$:

$$\forall i \in [0,\ldots,k] \quad \chi_i = [\lambda_i, \lambda_{i+1}[\tag{2.16}$$

avec $\lambda_1 = 0$ et $\lambda_{k+1} = +\infty$.

Si nous nous intéressons à présent aux régions des partitions, nous pouvons représenter la famille de partitions $A(I) = (P_\lambda)_{\lambda \in \mathbb{R}^+}$ par un ensemble de régions $H = \{x_i\}_{i \in 1\ldots m}$ où chaque région x_i appartient à suite continue de partitions P_λ caractérisées par leurs **échelles d'apparition**. On définit l'échelle d'aparition d'une région x comme la plus petite échelle telle que x appartient à une partitions P_λ :

$$\lambda^+(x) = min\{\lambda \in \mathbb{R}^+ | x \in P_\lambda\} \tag{2.17}$$

Notons que si $x \subset x'$, x et x' appartiennent à deux hiérarchies P_λ et $P_{\lambda'}$ avec $\lambda \leq \lambda'$. On a donc :

$$\forall (x,x') \in H^2 \quad x \subset x' : \lambda^+(x) \leq \lambda^+(x') \tag{2.18}$$

Si nous supposons que $P_{+\infty}$ ne contient qu'une seule région englobant la totalité de l'image, l'ensemble de régions $H = \{x_i\}_{\{i \in 1\ldots m\}}$ de $(P_\lambda)_{\lambda \in \mathbb{R}^+}$ définit une hiérarchie de P_0 sur I. En effet :

(i) H ne contient aucune région vide,

(ii) $I = P_{+\infty} \in H$ et

(iii) chaque singleton $x_i \in P_0$,

H répond donc bien aux 3 conditions de la définition 42 et est une hiérarchie. De plus λ^+ vérifiant l'équation 2.18 il est un *critère croissant* pour l'inclusion.

Une région x apparaissant au niveau $\lambda^+(x)$ survit dans la hiérarchie jusqu'à ce qu'elle soit fusionnée avec d'autres régions de son voisinage pour former son père $p(x)$. L'échelle où cet évènement se produit est appelée **l'échelle de disparition** de x et est notée $\lambda^-(x)$:

$$\lambda^-(x) = \min_{x' \in H, x \subset x'} \lambda^+(x') = \lambda^+(p(x)) \tag{2.19}$$

(H, λ^+) est donc **une hiérarchie indicée**, les partitions renvoyées par A correspondant aux coupes naturelles (Déf. 49) de cette hiérarchie. Dans ce type de représentation ensemble-échelle ce n'est plus la partition mais la région qui apparaît comme l'élément conceptuel. Chaque région sera ainsi caractérisée par son échelle d'apparition λ^+ et son échelle de disparition λ^- constituant son *intervalle de vie* autrement appelé **intervalle de persistance**.

$$\chi^*(x) = \{\lambda | x \in C^*_\lambda(H)\} = [\lambda^+(x), \lambda^-(x)[\tag{2.20}$$

2.4.2 Pyramides régulières

Une pyramide régulière est définie comme une séquence d'images dont la taille diminue de façon exponentielle [TANIMO75, BURT83, JOLI94, ROZENF88, BISTER90, KONIK95, PYZLOT95, CHEN95, OVERTU95, CINQU95, KONIK96, LOZANO98, ROSENF98]. Chacune des images de cette séquence correspond à un *niveau* de la pyramide avec le premier niveau correpondant à l'image originale tandis que le dernier niveau, le plus élevé, est en général composé d'un unique pixel dont la valeur est une moyenne pondérée des autres pixels composant l'image d'origine.

En utilisant les relations de voisinage définies sur l'image, la *fenêtre de réduction* d'un pixel de la pyramide le relie à un ensemble de pixels connexes dans l'image de niveau précédent. L'ensemble des pixels appartenant à une fenêtre de réduction est appelé les *enfants* ou les fils du pixel définissant celle-ci. Un tel pixel est, quant à lui, appelé le *père* des pixels appartenant à la fenêtre de réduction. On peut étendre cette relation père/enfant par transitivité, à n'importe quel niveau de la pyramide.

On appelle *champ récepteur* d'un pixel, l'ensemble de ses enfants définis dans l'image de base, au premier niveau de la pyramide. Toutes les valeurs calculées par le biais de la pyramide régulière sont stockées à un niveau particulier de la pyramide. Chaque pixel de la pyramide aura une fenêtre de réduction de taille et de cardinal identique.

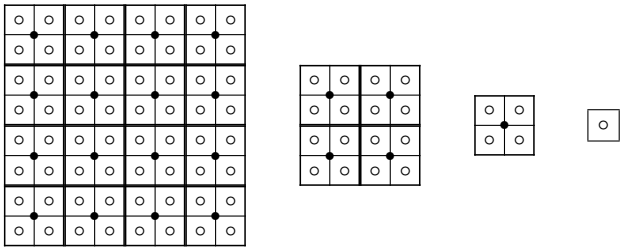


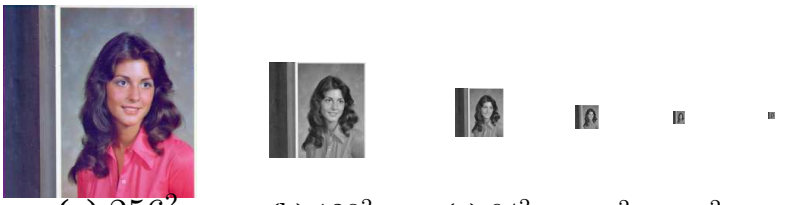
FIG. 2.28 – Une pyramide régulière $2 \times 2/4$

On appelle *facteur de réduction*, le rapport entre la taille de deux images successives de la pyramide. Pour les pyramides régulières, ce facteur reste constant entre tous les niveaux consécutifs. Le contenu de chaque pixel est calculé par le biais d'une *fonction de réduction*, par rapport au contenu des ses fils appartenant à sa fenêtre de réduction.

Pour définir formellement les pyramides régulières, on utilise le rapport $N \times N/r$ où $N \times N$ représente la taille de la fenêtre de réduction tandis que r représente le facteur de réduction. On distingue différents types de pyramides en fonction de ce rapport :

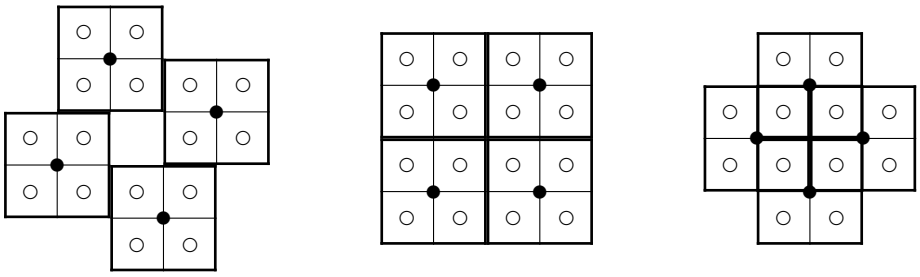
- si $N \times N/r < 1$, la pyramide est appelée une *Pyramide trouée et non recouvrante*. Dans le cadre de ces pyramides, certains pixels ne possèdent pas de père [KROPAT98] (voir par exemple le pixel central sur la figure 2.30(a)) ;
- si $N \times N/r = 1$, la pyramide est appelée une *pyramide non recouvrante non trouée* (voir par exemple la figure 2.30(b)) ;
- si $N \times N/r > 1$, la pyramide est appelée une *pyramide recouvrante*. Chaque pixel d'une telle pyramide a plusieurs *pères potentiels* [BURT81]. Si chaque enfant sélectionne un père parmi ces pères potentiels, l'ensemble des enfants de la fenêtre de

réduction de chaque pixel est réorganisé. Le champ récepteur d'un pixel peut dans ce cas prendre n'importe quelle forme incluse dans le champ récepteur original.



(a) 256^2 (b) 128^2 (c) 64^2 (d) 32^2 (e) 16^2 (f) 8^2

FIG. 2.29 – Une $2 \times 2/4$ pyramide. La fonction de réduction est une gaussienne centrée sur le pixel survivant. La taille de chaque image est indiquée en dessous de celle-ci.



a) $2 \times 2/5$ b) $2 \times 2/4$ c) $2 \times 2/2$

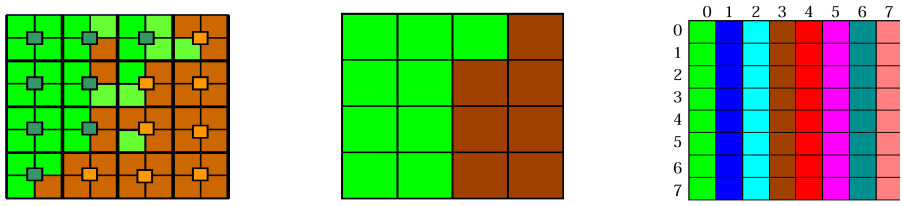
FIG. 2.30 – Trois types de pyramides régulières

Le principales propriétés des pyramides régulières ont été énumérées par Bister [BISTER90] :

- Les algorithmes utilisés sont indépendants de la résolution des régions recherchées.
- Ce type de structure permet de réduire l'influence du bruit.
- Les propriétés globales sont converties en propriétés locales.
- Les temps de calcul sont diminués par l'utilisation du principe " diviser pour conquérir ".
- Les algorithmes peuvent être implémentés et tirer parti des architectures parallèles.
- la détermination de certaines régions d'une image à faible coût en utilisant des images de faibles résolutions.

Malgré toutes ces propriétés intéressantes, ce type de structure hiérarchique présente plusieurs limitations dues essentiellement à la taille limitée et à la forme fixe de la fenêtre de réduction. Ces contraintes alliées à un facteur de réduction constant ne permettent pas aux pyramides régulières de s'adapter facilement à la variabilité des données. On peut en effet relever plusieurs problèmes induits par la rigidité de cette représentation tels que la non stabilité des données lors de légères variations de l'image. Les pyramides régulières ne sont , de ce fait, pas robustes à des changements d'échelle ainsi qu'aux décalages et rotations. Le fait qu'elles ne permettent de coder qu'un nombre limité de régions à certains niveaux lui interdit le codage de régions allongées [BISTER90] (régions occupant

beaucoup plus de pixels suivant une direction de l'image que l'autre). Cette dernière limitation est illustrée sur la figure 2.31(c) représentant le niveau de base d'une pyramide $2 \times 2/4$ d'une image composée de 8 régions allongées,composées de 8 pixels chacune. La décroissance de ce type de pyramide étant de 2 aussi bien en longueur qu'en largeur, la partition ne peut être décrite correctement qu'au niveau 3 de la pyramide. Or il ne reste que 4 pixels à ce niveau, ce qui interdit de coder correctement ce type de configuration. Notons enfin que les pyramides régulières ne peuvent en aucun cas garantir la préservation de la convexité comme l'illustrent les Figures 2.31(a) et (b). Les niveaux 1 et 2 de cette $2 \times 2/4$ pyramide sont représentés sur la figure 2.31(a), chaque pixel de niveau 2 étant représenté au centre de sa fenêtre de réduction. Le niveau 1 de l'image est composé de pixels de 2 classes (marron et vert), l'affectation d'un pixel de niveau 2 à une classe étant effectué par une décision à la majorité dans sa fenêtre de réduction. Comme on peut le voir sur la figure 2.31, cette règle fournit deux régions connexes au niveau 2 alors que la décomposition en région des classes vertes et marrons au niveau 1 n'est pas connexe. On ne peut donc pas inférer la connexité de régions à la base de la pyramide à partir de celles définies à de plus hauts niveaux.



(a) (b) (c)

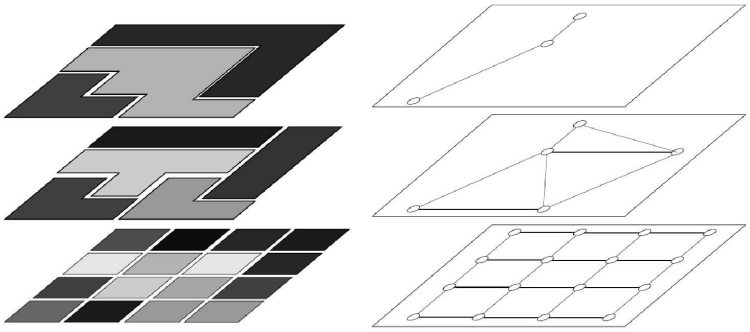
FIG. 2.31 – niveau de base et niveau 1 sur une image 8×8 (a) et segmentation de l'image en fonction des valeurs de pixels de niveau 1(b) (c) niveau de base d'une pramide $2 \times 2/4$ pour une image 8×8

Notons également que les transformations en ondellettes discrètes [VAUTRO96, MALLAT96] ont un principe quelque peu similaire. L'image est en effet décomposéee en 4 sous-images représentant les détails (hautes fréquences horizontales, verticales et diagonales) et les basses fréquences. Les 4 pyramides sont obtenues recursivement par filtrage de l'image basse fréquence obtenue. De ce point de vue les pyramides régulières peuvent être considérées comme un cas particulier de représentation échelle/signal. Utilisées en détection et en segmentation, les pyramides régulières le furent également en codage d'image [HUFFMA52]. Des applications en compression d'image, réduction de bruit, détection d'arêtes, flot optique et stéréo-vision montrent le fort potentiel de ce courant des représentations hiérarchiques [MALLAT96].

2.4.3 Pyramides irrégulières

Les pyramides irrégulières, également appelées pyramides de graphes, permettent de s'affranchir des limitations des pyramides régulières mentionnées dans la section 2.4.2 tout en préservant leurs principaux avantages. La pyramide irrégulière doit son nom au type de voisinage particulier des cellules qui la composent : le nombre de voisins de chaque cellule n'est pas fixe, il dépend de chaque cellule et est donc non régulier. Sa particularité

principale réside dans le fait que la forme des régions extraites n'est contrainte par aucun critère géométrique.



(a)

FIG. 2.32 – Une pyramide irrégulière basée sur une grille 4-connexe (a) et le RAG correspondant (b).

La contraction (Def. 12) est l'opération privilégiée pour construire les pyramides. Toutefois, la contraction d'une boucle étant une opération interdite, l'ensemble des arêtes contractées pour passer d'un niveau au niveau supérieur de la pyramide ne doit pas comporter de cycles, c'est donc une forêt. Dans le cadre des pyramides irrégulières un tel ensemble est appelé un **noyau de contraction**.

Définition 52 Noyau de contraction

Un noyau de contraction $K \subset E$ d'un graphe $G = (V, E)$ est une forêt de G codant un ensemble de sommets à fusionner.

Dans le cadre du codage de partition, chaque arbre d'un noyau de contraction représente la fusion d'un ensemble connexe de régions en une seule région au niveau supérieur. Un tel ensemble est appelé la **fenêtre de réduction** du sommet de niveau supérieur.

Définition 53 Fénêtre de réduction d'un sommet

Soit $P = (G_0, \ldots, G_n)$ une pyramide de graphes K_l, $l \geq 1$ un noyau de contraction permettant de construire G_l à partir de G_{l-1}. Chaque composante connexe $\tau \subset K_l$ de K_l code la contraction d'un ensemble connexe de sommets de G_{l-1} en un seul sommet v_τ de G_l.

L'ensemble $\{v_1, \ldots, v_p\}$ des sommets de G_{l-1} incidents à τ est appelé la fenêtre de réduction de v_τ. Le sommet v_τ est le père des sommets $\{v_1, \ldots, v_p\}$ qui sont les fils de v_τ.

Définition 54 Champ récepteur d'un sommet

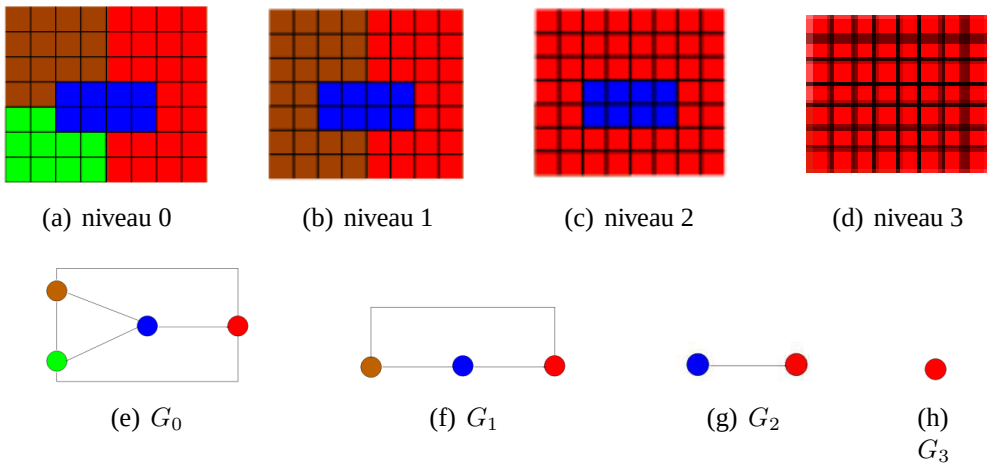
Le champ récepteur d'un sommet représente un ensemble de sommets définis dans le graphe initial. Il est obtenu en itérant la relation père/fils induite par la fenêtre de réduction d'un sommet.

Notez que les définitions précédentes sont la traduction dans le cadre des pyramides irrégulières des concepts équivalents introduits dans les pyramides régulières (Section 2.4.2).

2.4.3.a Pyramides de graphes simples

Dans les pyramides irrégulières , la notion de graphe intervient pour modéliser une relation entre les différentes régions du partitionnement. Le graphe d'adjacence, noté RAG, défini dans la section 2.3.1, est la structure de base à partir de laquelle vont être extraits des sous-graphes, sur des critères qualitatifs et quantitatifs lors de la construction des niveaux successifs de la pyramide (figure 2.32).

Ces pyramides sont donc définies comme une pile de graphes $(G_1, \ldots, G_n)$ successivement réduits (figure 2.33).



(a) niveau 0 (b) niveau 1 (c) niveau 2 (d) niveau 3

(e) G_0 (f) G_1 (g) G_2 (h) G_3

FIG. 2.33 – (a)(b)(c)(d)les 4 premiers niveaux d'une pyramide irrégulière basée sur une grille 4-connexes sur une image 8×8 avec (e)(f)(g)(h) le graphe correspondant à chaque niveau

Le graphe initial $G_0 = (V_0, E_0)$ peut être défini par une grille régulière en associant chaque pixel d'une image à un sommet de V_0, les arêtes E_0 représentant les relations d'adjacences entre ces pixels. Dans le cadre des pyramides de graphes simples on désigne usuellement un sommet particulier de chaque arbre d'un noyau de contraction. Ce sommet est appelé un sommet survivant et le noyau de contraction est noté (S, K), où S représente l'ensemble des sommets survivants et K l'ensemble des arêtes à contracter.

Comme nous l'avons vu dans la section 2.1, l'opération de contraction ne préserve pas obligatoirement la simplicité du graphe. Le processus permettant de passer d'un graphe G_l à G_{l+1} dans le cadre des pyramides de graphes simples va donc s'effectuer en quatre étapes :

1. Sélectionner un ensemble de sommets survivants,
2. Sélectionner un ensemble d' arêtes connectant chaque sommet non survivant à un sommet survivant.
3. Contracter cet ensemble d'arêtes.
4. Supprimer toutes les arêtes multiples et les boucles éventuellement créées lors des opérations de contraction.

La sélection des sommets survivants (étape 1) ainsi que celle de l'ensemble des arêtes à contracter (étape 2) est dépendante de la méthode de décimation utilisée et sera étudiée plus en détail dans la Section 2.6.

2.4.3.b Pyramides de graphes duaux

Les pyramides de graphes simples, définies dans la Section 2.4.3.a , permettent de s'affranchir des principales limitations des pyramides régulières tout en préservant leurs bonnes propriétés. Cependant, comme nous l'avons décrit dans la Section 2.3.1, l'utilisation de graphes simples dans le cadre du codage de partition implique de nombreuses limitations : L'absence de boucles ne permet pas de caractériser les relations d'inclusion et l'utilisation d'une seule arête entre deux sommets réduit l'information de frontière à une simple information d'adjacence (l'existence d'au moins une frontière commune entre deux régions, Def. 31).

Les pyramides de graphes duaux ont été introduites pour pallier aux principaux inconvénients des pyramides de graphes simples dans le cadre du codage de partitions. Les opérations de réduction pour passer d'un graphe G_l à G_{l+1} sont définies, comme dans les pyramides de graphes simples à l'aide d'un noyau de contraction (que l'on note ici simplement K). Le graphe obtenu peut être non simple et contenir soit des boucles soit des arêtes multiples entre des sommets. Toutefois toutes les arêtes multiples et toutes les boucles ne codent pas des informations pertinentes pour la partition. Plus précisément, les boucles vides (Def. 17) et les arêtes doubles (Def. 18) correspondent respectivement à des frontières internes de régions et à des divisions artificielles de courbes frontières. Ces deux types d'arêtes sont supprimés à l'aide d'un **noyau de suppression**.

Définition 55 Noyau de suppression

Un noyau de suppression $\overline{K}$, d'un graphe planaire G, est une forêt de $\overline{G}$ telle que toute arêtes de $\overline{K}$ est incidente à un sommet de $\overline{G}$ de degré 1 ou 2.

Dans le cadre des graphes duaux, la construction d'un nouveau niveau à partir d'un graphe G_l itère donc les étapes suivantes :

1. Construction d'un noyau de contraction K de G_l,
2. Contraction de l'ensemble des arête de K dans G_l et suppression des arêtes correspondantes dans $\overline{G_l}$. Appellons G' le graphe obtenu.
3. Définition d'un noyau de suppression $\overline{K}$ de G' incident à tous les sommets de degré inférieur à 2 de $\overline{G'}$.
4. Suppression dans G' des arêtes de $\overline{K}$ et contraction des arêtes correspondantes dans $\overline{G'}$. La paire de graphes obtenue est notée $(G_{l+1}, \overline{G_{l+1}})$.

La construction du graphe G' est considérée comme un niveau a part entière par certains auteurs. Cette convention est utilisée dans la figure 2.34 qui illustre le mécanisme de construction précédent.

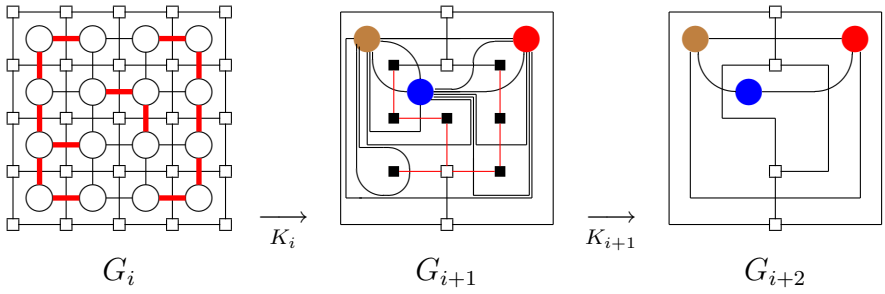


FIG. 2.34 – Une étape de la construction d'une pyramide de graphes duaux.

2.4.3.c Pyramides irrégulières bornées

Marfil [MARFIL04, MARFIL07] a récemment défini un nouveau modèle pyramidal, appelé *pyramide irrégulière bornée*, présenté comme étant un mélange entre les pyramides régulières et irrégulières, permettant de conserver les avantages liés à ces deux structures. Le principe des pyramides irrégulières bornées est d'utiliser une approche régulière sur les larges zones homogènes de l'image et d'utiliser une approche irrégulière sur les zones de géométrie plus complexe en combinant une structure régulière $2 \times 2/4$ (Figure 2.28) avec un graphe simple.

Cette combinaison de structures régulières et irrégulières donne une hiérarchie de graphes dans laquelle chaque niveau $G_l = (N_l, E_l)$ est constitué d'un ensemble de nœuds N_l liés par un ensemble d'arêtes intra-niveau E_l.

Il y a deux types de nœuds :
– les *nœuds réguliers*, appartenant à la structure régulière $2 \times 2/4$ (Figure 2.35(a))
– les *nœuds virtuels* appartenant à la structure irrégulère (Figure 2.35(b))

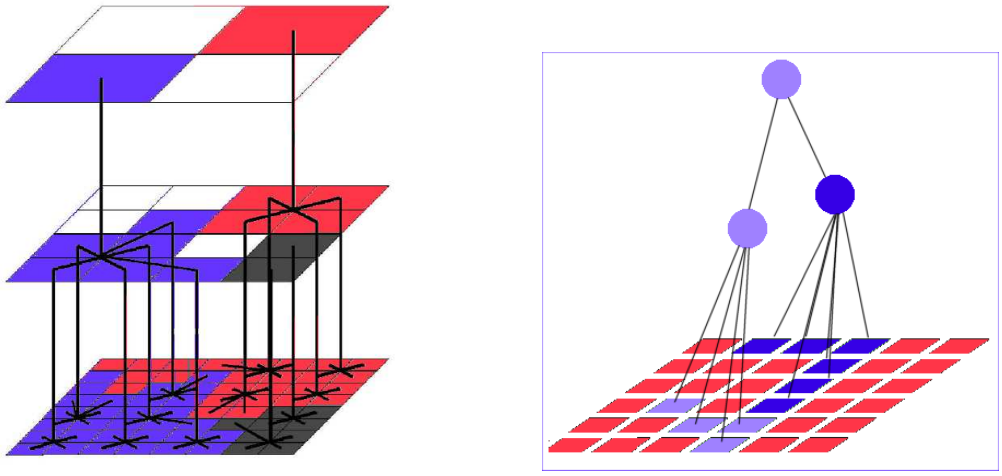
Deux nœuds $n_i \in N_l$ et $n_j \in N_l$ qui sont voisins au niveau l sont reliés par une arête intra-niveau $e_{ij} \in E_l$.

Le niveau de base G_0 de cette hiérarchie correspond au graphe des pixels de l'image à traiter, défini en 8-connexité. Chacun des nœuds de ce graphe est un nœud régulier.

Le processus pour construire le graphe G_{l+1} à partir de G_l s'effectue en trois étapes :

1. procéder à une décimation régulière
2. appliquer une stratégie union find sur les régions restantes
3. établir les liaisons intra-niveaux

La première étape va donc consister à regrouper les pixels par quatre de façon régulière, s'ils répondent à un critère d'homogénéité à définir. Il faut ensuite les relier par des arêtes inter-niveaux pour coder les relations pères/fils. Les étapes suivantes vont utiliser une stratégie de regroupement utilisée par Brun et Kropatsch [BRUN03B], appelée *union-find*, pour relier les pixels non traités lors de la première étape. L'algorithme union-find a été proposé par Tarjan [TARJAN75] comme une méthode générale permettant de partitionner un ensemble de classes disjointes. Son nom tient au fait qu'il permet d'effectuer deux opérations, **find** permettant de déterminer la classe d'un élement (utile pour déterminer si deux élements sont de même classe) et **union** permettant de réunir plusieurs classes. Ce type d'algorithme utilise en général une structure arborescente pour représenter les



(a) 2 niveaux d'une pyramide irrégulière bornée

(b) préservation de la connexité

FIG. 2.35 – (a) Noeuds réguliers de la pyramide avec les liaisons inter/intra niveaux. Les regroupements sont ici effectués sur critère colorimétrique. Les régions blanches ne sont pas homogènes. (b) Mélange de noeuds réguliers/non-réguliers formant un exemple de pyramide irrégulière bornée, illustrant la préservation de la connexité.

ensembles. Chaque noeud contient une référence vers son père qui représente une classe. Si deux nœuds appartenant à deux ensembles (ou arbres) différents sont jugés similaires, l'opération d'union est effectuée en affectant la racine d'un des deux arbres comme étant le père de l'autre.

Dans le cadre des pyramides irrégulières bornées, ce procédé va relier deux nœuds respectant un critère de similarité, ce qui donne :

- un nœud régulier si les deux nœuds sont réguliers
- un nœud virtuel de niveau $l+1$ si les deux nœuds de niveau l sont virtuels
- un nœud virtuel si nous avons relié un nœud virtuel avec le parent d'un nœud régulier voisin.

L'utilisation de ce type de structure met en évidence plusieurs avantages. L'utilisation d'une structure présentant des aspects réguliers améliore les temps de calculs par rapport à une structure irrégulière. Contrairement aux pyramides régulières, les pyramides régulières bornées permettent la préservation de la connexité. Cette propriété est assurée par le fait qu'un champ récepteur (Déf.54) est toujours généré (ou étendu) en reliant exclusivement des nœuds voisins(Figure 2.35(b)). Il est également possible de représenter des régions allongées en utilisant les graphes simples inclus dans cette structure.

2.4.3.d Pyramides de cartes combinatoires

Les pyramides combinatoires ont été définies par Brun et Kropatsch [BRUN02] pour pallier aux limitations des pyramides de graphes duaux tout en conservant leurs principaux avantages. Une pyramide combinatoire est définie comme une pile de cartes combinatoires successivement réduites par une séquence d'opérations de contraction et de suppression.

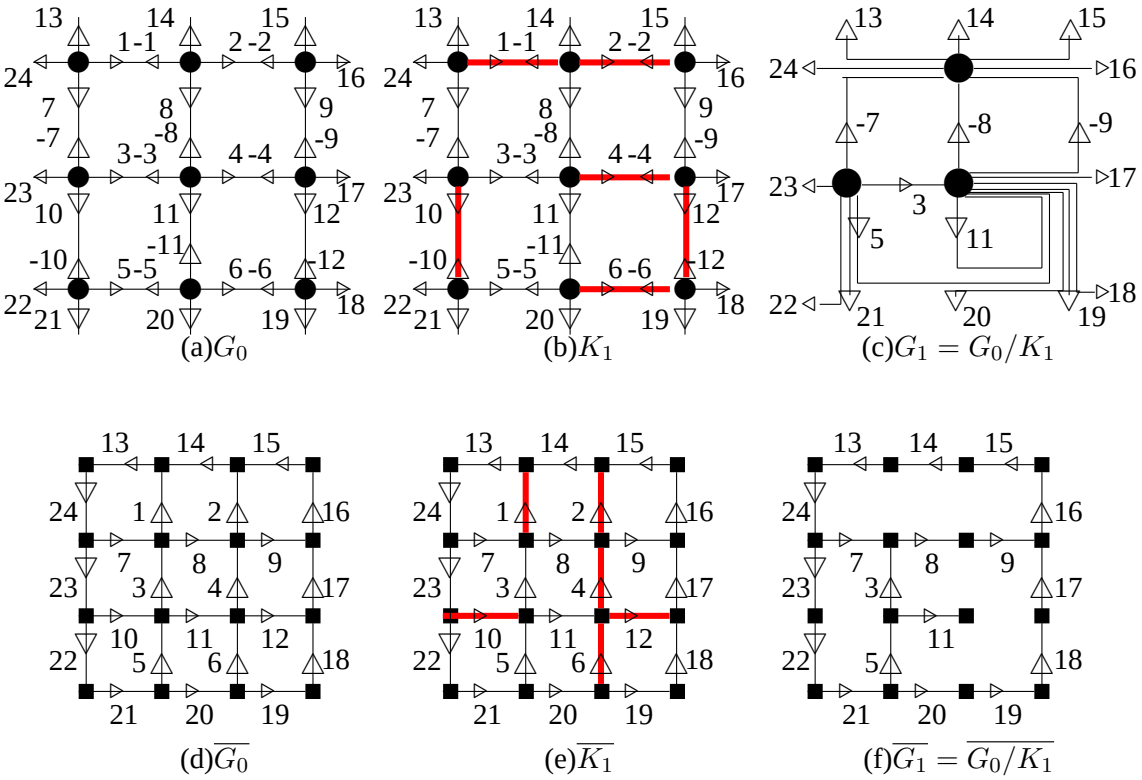


FIG. 2.36 – Contraction d'une grille 3×3 par un noyau de contraction K_1 dans la carte initiale et son dual. Les brins négatifs ne sont représentés dans les cartes duales (d-f) mais se déduisent facilement puisque sur cette figure deux brins de signe opposés sont incidents aux deux sommets d'une même arête.

De la même façon que nous l'avons défini pour les graphes dans la section 2.4.3, il est nécessaire d'introduire la notion de noyau de contraction/suppression dans le cadre des cartes combinatoires.

Définition 56 Noyau de Contraction

Soit une carte combinatoire $G = (\mathcal{D}, \sigma, \alpha)$. Un sous-ensemble symétrique (Def. 35) de brins $K \subset \mathcal{D}$ sera appelé un noyau de contraction si et seulement si :

- *K forme une forêt de G (Définition 6) ;*
- *K ne contient pas tous les brins de G :*

$$\mathcal{BS} = \mathcal{D} - K \neq \emptyset.$$

L'ensemble $\mathcal{BS}$ est appelé l'ensemble des brins survivants.

Cette définition correspond à celle donnée dans le cadre des graphes (Def. 56) avec la subsitution des arêtes par des ensembles symétriques de brins. De plus les cartes étant définies à partir des arêtes, il n'est pas possible de coder un sommet isolé sans arête incidente à l'aide de carte. On exige donc que l'ensemble des arêtes à contracter soit strictement plus petit que celui de la carte initiale. Une exemple de contraction opérée par un noyau est représenté sur la figure 2.36.

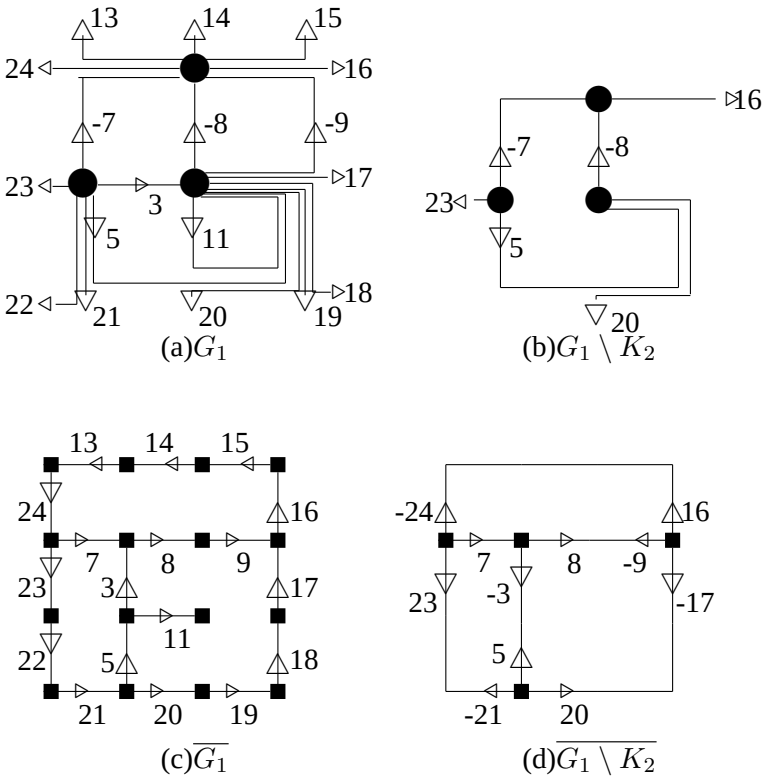


FIG. 2.37 – Simplification d'une grille 3×3 contracté (Figure 2.36) par un noyau de suppression K_2 dans la carte initiale et son dual. Le brin -7 sur le sommet opposé au brin 7 n'a pas été représenté sur la figure (d) pour ne pas la surcharger.

En reprenant le même schéma que pour la définition 38 permettant de définir les contractions par rappport au dual d'une carte, le noyau de suppression peut se définir comme un noyau de contraction appliqué à la carte duale.

Définition 57 Noyau de Suppression

Soit une carte combinatoire $G = (\mathcal{D}, \sigma, \alpha)$. Un sous-ensemble symétrique de brins $K \subset \mathcal{D}$ sera appelé un noyau de suppression s'il forme un noyau de contraction de $\overline{G}$.

Notez qu'une opération de contraction dans le graphe dual est équivalente à une opération de suppression dans le graphe primal (d'où le nom de noyau de suppression).

Cette définition revenant à appliquer un noyau de contraction dans la carte duale elle assure la préservation de la connexité dans cette carte. Comme la carte et son dual ont le même nombre de composantes connexes [GARETH78, BRUN99], un noyau de suppression préserve également la connexité de la carte initiale.

La dualité entre ces définitions de noyaux permet de rester libre sur le type de graphe à utiliser pour le codage de partition. On utilisera donc un noyau de contraction pour fusionner les régions si l'on utilise un codage de la partition avec une carte correspondant à un graphe de régions (Def. 19), le noyau de suppression ne servant qu'à supprimer les arêtes redondantes. Inversement nous utiliserons un noyau de suppression pour fusionner les régions si la partition est codée à l'aide d'une carte des frontières(Def. 20), l'opération de contraction n'étant utilisée que pour supprimer les arêtes redondantes.

Si la carte G correspond à un graphe de régions, les arêtes redondantes sont caractérisées dans $\overline{G}$ et correspondent :

- Soit à des arêtes incidente à un sommet de degré 1 dans $\overline{G}$ (arête $(11, -11)$, Fig. 2.37(c)). Ces arêtes correspondent à des boucles d'intérieur vide dans G (boucle $(11, -11)$, Fig. 2.37(a)) et sont appelées des **boucles vides**. Si b est un brin d'une boucle vide appartenant au sommet de degré 1 on a $\varphi(b) = b$. Cette dernière relation caractérise la boucle vide.
- Soit à des arêtes incidente à un sommet de degré 2 dans $\overline{G}$ (e.g. sommet $(-8, 9)$, Fig. 2.37(c)). Ces arêtes correspondent à des faces d'intérieur vide dans G (face $(-8, 9)$, Fig. 2.37(a)) et sont appelées des **arêtes doubles**. Les brins des arêtes doubles peuvent s'interpréter comme des frontières dans la carte de base (voir par exemple la suite $20.19.18.17$, Fig. 2.36(f) de brins appartenant à des arêtes doubles) et sont donc également appelés des **brins frontière**. Si b est un brin frontière, il est incident à un sommet de degré 2 et l'on a $\varphi^2(b) = b$

Dans la suite de ce document les cartes combinatoires coderont des cartes de régions et les arêtes redondantes définissant les noyaux de suppression seront définies comme indiqué ci-dessus. De plus afin de distinguer les brins frontières nous décomposeront les noyaux de suppressions en deux sous noyaux :

Définition 58 Noyau de Suppression de boucles vides

Soit $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ une carte. Un noyau de suppression de boucle vide sur G est un noyau de suppression K sur G tel que :

$$\forall b \in K \quad \varphi(b) = b \text{ ou } \varphi(\alpha(b)) = \alpha(b)$$

Définition 59 Noyau de Suppression de brins frontières

Soit $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ une carte. Un noyau de suppression de brins frontières sur G est un noyau de suppression K sur G tel que :

$$\forall b \in K \quad \varphi^2(b) = b$$

Sur la Fig. 2.37 le noyau de suppression K_2 défini par :

$$K_2 = \{11, -11, 24, -13, 13, -14, 14, -15, -16, 17, -18, 18, -19, 19, \\ -20, 21, -22, 22, -23, 3, -5, -8, 9\}$$

est décomposé en deux sous noyaux :

- Un noyau de suppression de boucles vides :

$$K_2^a = \{11, -11\}$$

- Un noyau de suppression de brins frontières :

$$K_2^b = \{24, -13, 13, -14, 14, -15, -16, 17, -18, 18, -19, 19, \\ -20, 21, -22, 22, -23, 3, -5, -8, 9\}$$

Dans la suite de cet exposé un noyau de suppression désignera donc indifféremment un noyau de suppression de boucles vides ou un noyau de suppression de brins frontières.

Soient une carte initiale $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ et un noyau de suppression K_s donnant une fois appliqué la carte réduite $G' = (\mathcal{BS} = \boldsymbol{\mathcal{D}} - K, \sigma', \alpha)$. Brun [BRUN02] montre que la permutation σ' de G' peut se définir par :

$$\forall d \in \mathcal{BS} \quad \sigma'(d) = \sigma^n(d) \text{ avec } n = Min\{k \in \mathbb{N}^* \mid \sigma^k(d) \in \mathcal{BS}\}. \tag{2.21}$$

et de part la dualité des opérations de contraction/suppression, on a dans le cas d'une opération de contraction :

$$\forall d \in \mathcal{BS} \quad \varphi'(d) = \varphi^n(d) \text{ avec } n = Min\{k \in \mathbb{N}^* \mid \varphi^k(d) \in \mathcal{BS}\}. \tag{2.22}$$

ou encore, puisque $\varphi'(\alpha(d)) = \sigma'(\alpha^2(d)) = \sigma'(d)$:

$$\forall d \in \mathcal{BS} \quad \sigma'(d) = \varphi^n(\alpha(d)) \text{ avec } n = Min\{k \in \mathbb{N}^* \mid \varphi^k(\alpha(d)) \in \mathcal{BS}\}. \tag{2.23}$$

Étant donné un noyau de suppression (resp. contraction) K, nous allons pouvoir définir la notion de **chemin de connexion**, comme étant, pour un brin survivant b, la séquence de brins non survivants qu'il faut traverser pour atteindre le σ (resp. φ) successeur de b dans la carte réduite. Cette notion de chemin de connexion peut être interprétée comme l'analogue en termes de brins des fenêtres de réduction (Section 2.4.3).

Définition 60 Chemins de connexion

Soient une carte combinatoire $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$, un noyau K et un brin $b \in \mathcal{BS}$. Le chemin de connexion (ou connecting walk) associé à b est défini par :

– *Si K est un noyau de contraction*

$$CW(b) = b\varphi(b)\ldots\varphi^{n-1}(b) \textit{ avec } n = Min\{k \in \mathbb{N}^* \mid \varphi^k(b) \in \mathcal{BS}\};$$

– *Si K est un noyau de suppression*

$$CW(b) = b\sigma(b)\ldots\sigma^{n-1}(b) \textit{ avec } n = Min\{k \in \mathbb{N}^* \mid \sigma^k(b) \in \mathcal{BS}\}.$$

Pour un brin $b \in G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ survivant, tel que $CW(b) = b.b_1 \ldots .b_p$, cette définition nous permet d'obtenir après l'application d'un noyau K, ses φ-successeurs et σ-successeurs dans la carte réduite $G' = (\mathcal{BS} = \boldsymbol{\mathcal{D}} - K, \sigma', \alpha)$:

$$\begin{cases} \varphi'(b) = \varphi(b_p) & \text{Si } K \text{ est un noyau de contraction} \\ \sigma'(b) = \sigma(b_p) & \text{Si } K \text{ est un noyau de suppression} \end{cases} \tag{2.24}$$

Cette notion peut être étendue récursivement et une extension des chemins de connexion avec plusieurs noyaux de types différents définit une **séquence de connexion** :

Définition 61 Séquences de connexion

Soient une carte combinatoire $G_0 = (\mathcal{D}, \sigma, \alpha)$ et une séquence de noyaux de contraction ou de suppression $K_1 \ldots, K_n$. Les séquences de connexion (SC) sont définies par la construction récursive suivante :

$$\forall b \in \mathcal{D} \quad SC_0(b) = b.$$

Pour chaque niveau $i \in \{1, \ldots, n\}$ et pour chaque $b \in \mathcal{BS}_i$

- *Si K_i et K_{i-1} ont le même type :*

$$SC_i(b) = SC_{i-1}(b_1) \ldots . SC_{i-1}(b_p);$$

- *Si K_i et K_{i-1} ont des types différents :*

$$SC_i(b) = b_1 . SC^*_{i-1}(\alpha(b_1)) \ldots . b_p . SC^*_{i-1}(\alpha(b_p)).$$

Où $(b_1, \ldots, b_p)$ est égal à $CW_i(b)$ et $SC^*_i(b)$ désigne la séquence de connexion $SC_i(b)$ sans son premier brin. Les noyaux $K_0 = \emptyset$ et K_1 ont le même type par convention.

Deux noyaux de même type sont deux noyaux de contraction ou de suppression alors que deux noyaux de types différents correspondent à des opérations différentes.

Cette nouvelle définition permet d'étendre l'équation 2.24 dans le cas des séquences de connexion, ce qui autorise le calcul d'un niveau de la pyramide directement à partir de sa base grâce au théorème suivant (voir [BR00A] pour la démonstration) :

Théorème 1 *Soient une carte combinatoire $G = (\mathcal{D}, \sigma, \alpha)$ et une séquence de noyaux de contraction ou de suppression successifs $K_1, \ldots, K_n$. La séquence de connection d'un brin $b \in \mathcal{BS}_i$ définie au niveau $i \in \{1, \ldots, n\}$:*

$$SC_i(b) = b_1 . \ldots . b_p$$

vérifie :

- *Si K_i est un noyau de contraction :*

$$\begin{array}{l} \textit{Si } p = 1 \\ \qquad \varphi_i(b) = \varphi(b) \\ \textit{sinon} \\ \qquad \varphi_i(b) = \left\{ \begin{array}{ll} \varphi(b_p) & \textit{Si } b_p \textit{ est contracté} \\ \sigma(b_p) & \textit{Si } b_p \textit{ est supprimé.} \end{array} \right. \end{array} ;$$

- *Si K_i est un noyau de suppression :*

$$\begin{array}{l} \textit{Si } p = 1 \\ \qquad \sigma_i(b) = \sigma(b) \\ \textit{sinon} \\ \qquad \sigma_i(b) = \left\{ \begin{array}{ll} \varphi(b_p) & \textit{Si } b_p \textit{ est contracté} \\ \sigma(b_p) & \textit{Si } b_p \textit{ est supprimé.} \end{array} \right. \end{array} .$$

Ce théorème permet de calculer n'importe quel niveau de la pyramide à partir de la carte de base. Toutefois il nécessite, pour obtenir une carte d'un certain niveau, d'avoir calculé récursivement toutes les cartes de niveau inférieur, ce qui n'est pas efficace en temps de calcul. Le théorème suivant [BR00A] permet de calculer un niveau de la pyramide sans avoir besoin de calculer de façon explicite, l'ensemble des niveaux intermédiaires.

Théorème 2 *Soient une carte combinatoire $G = (\mathcal{D}, \sigma, \alpha)$ et une séquence de noyaux de contraction ou de suppression successifs $K_1, \ldots, K_n$. Pour toute séquence de connexion $\mathcal{SC}_i(b) = b.b_1 \ldots .b_p$ définie par $b \in \mathcal{BS}_i$ avec $i \in \{1, \ldots, n\}$ nous avons si $p > 1$:*

$$b_1 = \begin{cases} \varphi(b) & \text{Si } K_i \text{ est un noyau de contraction} \\ \sigma(b) & \text{Si } K_i \text{ est un noyau de suppression} \end{cases}$$

et

$$\forall j \in \{2, \ldots, p\} \quad b_j = \begin{cases} \varphi(b_{j-1}) & \text{Si } b_{j-1} \text{ est contracté} \\ \sigma(b_{j-1}) & \text{Si } b_{j-1} \text{ est supprimé.} \end{cases}$$

Le théorème 2 permet de parcourir une séquence de connexion, à partir de son brin initial, connaissant les opérations qui ont permis de réduire chacun de ses brins. Il est donc judicieux de mettre en place un codage implicite de la pyramide.

Définition 62 Codage implicite d'une pyramide

*Soit une pyramide définie par une carte initiale $G_0 = (\mathcal{D}, \sigma, \alpha)$ et n noyaux successifs $K_1, \ldots, K_n$. Un codage implicite de la pyramide est un triplet composé de la carte initiale G_0 et de deux fonctions **état** et **niveau** telles que :*

- *la fonction état est définie de $\{1, \ldots, n\}$ dans les deux états binaires {Contracté,Supprimé} et code le type de chaque noyau.*

$$\textbf{\textit{état}} \left(\begin{array}{lcl} \{1, \ldots, n\} & \rightarrow & \{\textit{Contracté}, \textit{BoucleVide}, \textit{BrinFrontière}\} \\ i & \mapsto & \begin{cases} \textit{Contracté si } K_i \textit{ est un noyau de contraction,} \\ \textit{BoucleVide si } K_i \textit{ est un noyau de suppression de boucles vides} \\ \textit{BrinFrontière si } K_i \textit{ est un noyau de suppression de brins frontières} \end{cases} \end{array} \right.$$

- *La fonction niveau code pour chaque brin de la carte initiale $G = (\mathcal{D}, \sigma, \alpha)$ le niveau maximal où il survit.*

$$\begin{array}{rcl} \textbf{niveau} : \mathcal{D} & \rightarrow & \{1, \ldots, n+1\} \\ b & \mapsto & \max\{i \in \{1, \ldots, n+1\} \mid b \in \mathcal{BS}_{i-1}\}. \end{array}$$

Étant donné les deux fonctions $\mathbf{état}$ et $\mathbf{niveau}$ les relations du Théorème 2 peuvent se réécrire de la façon suivante :

$$b_1 = \begin{cases} \varphi(b) & \text{Si } \mathbf{état}(\mathbf{niveau}(b)) = \text{Contracté} \\ \sigma(b) & \text{Sinon} \end{cases}$$

et

$$\forall j \in \{2, \ldots, p\} \quad b_j = \begin{cases} \varphi(b_{j-1}) & \text{Si } \mathbf{état}(\mathbf{niveau}(b_{j-1})) = \text{Contracté} \\ \sigma(b_{j-1}) & \text{Sinon.} \end{cases}$$

La définition 62 nous fournit donc un codage implicite d'une pyramide combinatoire, nécessitant en pratique simplement le codage du niveau de chaque brin de la carte initiale.

Elle permet ainsi de retrouver n'importe quel niveau de la pyramide à partir du niveau initial en un temps proportionnel au nombre de brins de la carte de base [BR00A].

Considérons une carte $G_i = (\mathcal{BS}_i, \sigma_i, \alpha_i)$ d'une pyramide combinatoire, un brin $b \in \mathcal{BS}_i$ et la séquence de connexion $\mathcal{SC}_i(b)$. Cette séquence est composées d'un premier brin survivant au niveau i (b lui même) suivi d'une suite de brins contractés ou supprimés avant le niveau i. Parmis ces brins non survivant les brins contractés et les boucles vides codent des frontières internes à la région codées par le sommet $\sigma_i^*(b)$. Les brins frontières codent quand a eux des éléments de frontière entre le sommet $\sigma_i^*(b)$ et $\sigma_i^*(\alpha_i(b))$. Par exemple, Sur la figure 2.37, la séquence de connexion du brin 8 au niveau 2 est égale à $\mathcal{SC}_2(8) = 8.2.9$ (Fig. 2.36(e)). Le brin contracté 2 code une frontière interne à la première ligne qui est codée par le sommet $\sigma_2^*(8)$ tandis que les brins frontières 8 et 9 codent des éléments de frontière entre la première ligne et la région en bas à droite codée par $\sigma_2^*(-9)$ (Fig. 2.37). L'ensemble des brins frontières d'une séquence de connexion peuvent être retrouvés directement à partir du codage implicite [BRUN03A]. Une telle séquence de brins est appellée la séquence de brins frontière (*Séquence of boundary darts*) du brin survivant.

Définition 63 ***Séquence de brins frontières***

Étant donné une pyramide $P = (G_0, \ldots, G_n)$, une carte $G_i = (\mathcal{BS}_i, \sigma_i, \alpha_i)$ et un brin $b \in \mathcal{BS}_i$, la séquence de brins frontières $SBD_i(b) = d_1 \ldots .d_p$ de b est définie par :

$$d_1 = b \text{ et } \forall j \in \{1, \ldots, p-1\} \quad d_{j+1} = \varphi_0^{n_j}(\alpha_0(d_j))$$

avec $n_j = Min\{k \in \mathbb{N}^* \mid \mathbf{niveau}(\varphi^k(\alpha_0(d_j))) > i$ *ou* $\mathbf{etat}(\mathbf{niveau}(\varphi^k(\alpha_0(d_j)))) = brinFrontiere\}$.

Par extension, on dira que la séquence de brins frontière d'un sommet $\sigma_i^*(b) = (d_1, \ldots, d_p)$ est définie comme la concaténation des séquences de brins frontières $SBD_i(d_1) \ldots SBD_i(d_d)$. Par exemple la séquence de brins frontières de $\sigma_2^*(8) = (8, 16, 7)$ (Fig. 2.37(d)) est égale à (Fig. 2.36(f)) :

$$SBD_2(8).SBD_2(16).SBD_2(7) = 8.9.16.15.14.13.24.7$$

Ce modèle pyramidal basé sur les cartes combinatoires se distingue de ceux basés sur les graphes simples ou duaux de par un codage implicite des sommets et un codage explicite de l'orientation. Cette structure bénéficie de tous les avantages inhérents à l'utilisation des cartes combinatoires (codage implicite du dual, caractérisation des relations d'inclusion, codage implicite de l'orientation) et la mise en correspondance des arêtes de la carte avec les frontières fournit une description fine de celle-ci. La possibilité d'utiliser le codage implicite de la pyramide précédemment défini, permet d'alléger la capacité de stockage nécessaire au maintien d'une telle structure tout en maintenant un accès rapide à chacun des niveaux.

2.5 Pyramides et minimisation d'énergies

2.5.1 Cadre énergétique

Dans cette section nous allons envisager la segmentation comme un problème de modélisation à part entière. Cette approche s'inscrit dans la lignée des approches par minimisation d'énergie. Les différentes théories proposées peuvent être classées en fonction de la nature des images qu'elles considèrent (continue ou discrète) mais également en fonction du modèle considéré, qui peut être déterministe ou stochastique. Les formulations *bayesiennes*, utilisées en traitement d'image considèrent généralement des modèles probabilistes avec des images discrètes. Au contraire, les méthodes variationnelles, utilisant les équations aux dérivées partielles, utilisent des images continues avec des modèles déterministes. Nous pouvons également citer les formulations par *codage minimal*, considérant les images discrètes, mais qui appliquent des modèles mixtes basés sur des composantes déterministes et stochastiques.

L'objectif de ces approches, face à une image observée I, est de sélectionner dans un ensemble de modèles possibles, celui qui modélise le mieux I. On parle en général de *problème inverse*, quand le modèle est considéré comme une cause des observations et que l'objectif est soit de «remonter» à cette cause, soit de le séparer d'autre facteurs perturbateurs (le bruit par exemple).

En appliquant le **principe de comparaison** énoncé par Koepfler et Morel [KOEPFL94] ce problème d'inférence de modèle peut se concevoir comme un problème d'optimisation. Ce principe énonce que dans un contexte déterminé, étant données deux segmentations différentes d'une même image, il est toujours possible de décider laquelle des deux peut être considérée comme meilleure que l'autre. Ce principe sous entend donc un ordre total sur l'ensemble des segmentations possibles. Cet ordre total implique l'existence d'une fonctionnelle E telle que si $E(P_i) < E(P_j)$, la segmentation P_i doit être considérée comme «meilleure» que P_j. Ceci nous amène à formuler le problème de segmentation, comme la spécification d'une fonction de coût E, appelée *énergie*, sur l'ensemble des segmentations possibles $\mathcal{M}$. La solution d'un problème de segmentation d'une image I revient ainsi à déterminer la segmentation minimisant cette énergie.

$$M^*(I) = \underset{M \in \mathcal{M}}{\operatorname{argmin}}\, E(M, I) \tag{2.25}$$

Le fait de rechercher un modèle, que nous pouvons interpréter comme une description simplifiée de l'image, soulève le problème du compromis à faire nécessairement entre la précision du modèle et sa simplicité. Dans cette optique, les thèories d'inférence de modèles proposent de décomposer l'énergie E en deux parties. Le terme supplémentaire porte uniquement sur le modèle et quantifie sa *complexité*. L'énergie à minimiser devient alors

$$E(M, I) = C(M) + D(M, I) \tag{2.26}$$

avec C une mesure de complexité de M et D une mesure de dissimilarité entre M et I.

Minimiser l'équation 2.26, revient à dire que pour deux partitions représentant de façon équivalente les données (même terme D), la partition la moins complexe (ou plus grossière) devra être préférée. Le rôle du terme de régularisation est d'éviter d'aboutir à la solution optimale, uniquement en terme d'attache aux données : la sur-partition absolue de l'image, dans laquelle chaque pixel est représenté par une région.

Comme nous venons de le voir, les énergies considérées dans le cadre du partitionnement d'image, se présentent sous la forme d'une opposition entre deux termes antagonistes. L'un concerne l'attache aux données, l'autre est considéré comme un terme de régularisation. Ces énergies peuvent donc être décrites par une **énergie affine** dans laquelle P est une partition du domaine de l'image et $\lambda \in \mathbb{R}^+$ est un paramètre servant à balancer les termes C et D en réglant leur importance relative.

Définition 64 Energie affine

*Une **énergie affine** est une énergie qui s'écrit*

$$E_\lambda(P) = \lambda C(P) + D(P) \tag{2.27}$$

avec P une partition du domaine des images et $\lambda \in \mathbb{R}^+$ un paramètre réglant l'importance entre les deux termes C et D.

Nous verrons dans la Section 2.5.2 que ce paramètre admet différentes interprétations possibles suivant la modélisation utilisée. L'appellation *énergie affine* vient du fait que pour une partition P fixée, $E_\lambda(P)$ sera vue comme une fonction de λ. Dans la majorité des approches, λ est considéré comme un *paramètre d'échelle*. Ceci s'explique par le fait qu'il joue globalement sur la finesse de la partition minimisant l'énergie considérée. L'échelle représente donc le compromis entre la *simplicité* et la *fidélité* du modèle.

Dans le cadre de partition, il est assez naturel de définir l'énergie de chaque région séparément et de définir l'énergie de la partition comme la simple somme des énergies de ses régions. Un tel type d'énergie est appelé une **énergie séparable**.

Définition 65 Énergie séparable

*Une **énergie séparable** d'une partition P est une énergie qui peut s'exprimer comme la somme des énergies portant sur les régions de P :*

$$E(P) = \sum_{R_i \in P} E'(R_i) \tag{2.28}$$

Nous utiliserons par la suite la même lettre pour désigner une fonction d'énergie définie sur une région et l'énergie séparable qui lui est associée, définie sur une partition.

2.5.2 Energie et modélisation

Nous allons illustrer les propos précédents par un tour d'horizon des principales approches énergétiques traitant du problème de partitionnement. Ces approches conduisent souvent à la résolution de problèmes mal posés. On dit d'un problème qu'il est mal posé, au sens d'Hadamard, s'il vérifie l'une de ces trois conditions (voir [POGGI85]) :

1. le problème n'a pas de solution,
2. la solution du problème n'est pas unique,
3. la solution du problème ne dépend pas continuement des données.

2.5.2.a Modélisations variationnelles

La nécessité d'introduire un terme de complexité de modèle dans les fonctionnelles trouve son origine dans les formalisations continues du problème de segmentation et plus exactement dans la théorie de la régularisation. En effet si l'on n'utilise que le terme de dissimilarité entre M et I le problème est *mal posé*. Dans ce cadre, le terme D est souvent appelé *terme d'attache aux données* et C *terme de régularisation*.

Mumford et Shah [MUMFOR89] proposent d'envisager le problème de segmentation d'images comme un problème d'approximation par une fonction lisse par morceaux, minimisant une fonctionnelle (équation 2.25). Une image I définie sur un domaine rectangulaire $\Omega \in \mathbb{R}^2$ est définie par un modèle $M : \Omega \rightarrow [0, 1]$ codant différentes régressions de I sur les régions d'une partition. Les contours de cette partition sont notés K.

$$E(M, K) = \iint_{\Omega} (M - I)^2 \partial x \partial y + \mu \iint_{\Omega} |\nabla M| \partial x \partial y + \nu Longueur(K) \tag{2.29}$$

Le premier terme traduit la qualité de la regression de I par M, le second basé sur le gradient va favoriser les intérieurs de région lisses alors que le troisième terme exprime une préférence pour les partitions comportant la plus faible longueur totale de contours. Les paramètres μ et ν servent à balancer l'importance de chacun de ces facteurs. Le fait de supprimer l'un de ces trois termes, dégénère le problème en un problème mal posé. C'est d'ailleurs le cas si l'on impose à l'image d'être constante par morceau. Le modèle M devient alors solution de :

$$E(M, K) = \iint_{\Omega} (M - I)^2 \partial x \partial y + \nu Longueur(K) \tag{2.30}$$

L'étude théorique de ce type de problèmes variationnels ainsi que leur résolution numérique ont été etudiées par Blake et Zisserman [BLAKE87] ainsi que Morel et Solimini [MOREL95].

Nous pouvons noter que l'énergie exprimée dans l'équation 2.31 est séparable. En effet le domaine Ω de l'image I peut se décomposer comme l'union des régions R_i et des frontières K de la partition en $\Omega = R_1 \cup \cdots \cup R_n \cup K$. Comme la mesure de longueur d'un courbe est additive pour l'union de courbes disjointes l'équation 2.31 peut se réécrire

$$\begin{aligned} E(M, K) = \sum_{i=1}^{n} \Big(& \iint_{R_i} (M - I)^2 \partial x \partial y + \frac{1}{2} \iint_{\partial R_i} (M - I)^2 \partial x \partial y \\ & + \mu \iint_{R_i} |\nabla M| \partial x \partial y + \frac{\nu}{2} Longueur(K_{R_i}) \Big) \end{aligned} \tag{2.31}$$

avec ∂R_i représentant la frontière de la région correspondante. Les facteurs $\frac{1}{2}$ que nous retrouvons pour les termes relatifs aux longueurs des frontières servent à éviter de comptabiliser deux fois une même frontière.

Nous avons donc pour toute région $R_i \in \Omega$:
– un terme d'attache aux données : $D(R_i) = \iint_{R_i}(M-I)^2 \partial x \partial y + \mu \iint_{R_i} |\nabla M| \partial x \partial y$
– un terme de régularisation : $C(R_i) = \frac{1}{2}\Big(\iint_{\partial R_i}(M-I)^2 \partial x \partial y + \nu Longueur(K_{R_i})\Big)$

Les approches par modèles de contours actifs [CHAN01] font également partie du domaine variationnel. Nous pouvons noter que ces approches utilisent également un terme d'attache aux données (gradient,variance, ...) ainsi qu'un terme de régularisation (normale, courbure, ...)

2.5.2.b Modélisation probabiliste

Dans les approches stochastiques de la segmentation d'image, cette dernière est vue comme la réalisation d'un processus aléatoire que l'on cherche à modéliser. Dans le cadre de la formulation Bayésienne, l'approche classique, proposée par Fisher [HEBERT95] est celle par *maximum de vraisemblance*. L'inférence bayésienne est ainsi nommée parce qu'elle fait un large usage de la règles de Bayes, elle-même conséquence d'une règle fondamentale du calcul de probabilités, appelée la règle du produit [CO63].

La principe de la formulation bayésienne est de trouver un modèle M qui permet de maximiser la probabilité $P_v(I|M)$ d'observer I sachant M. Ce modèle, à la base assez simple ne permet pas d'obtenir de solutions satisfaisantes. Il en ressort en effet que la solution triviale consistant à obtenir autant de régions que de pixels au départ apparait souvent comme la meilleure solution. Pour pallier à ce problème, la théorie de l'inférence bayésienne propose d'introduire une loi a priori P_p sur l'espace des modèles. Nous obtenons ainsi la probabilité a posteriori de M sachant I par :

$$P_P(M|I) = \frac{P_P(M)P_v(I|M)}{P_i(I)} \tag{2.32}$$

avec $P_i(I)$ représentant la probabilité d'observer l'image I parmi toutes les images possibles. Ce terme est généralement négligé.

La résolution peut alors s'effectuer par la stratégie du *maximum a posteriori*, qui revient à maximiser $P_P(M|I)$ ou encore à minimiser $-logP_P(M|I)$.

$$-logP_P(M) - logP_v(I|M) \tag{2.33}$$

Notons qu'encore une fois nous retrouvons une expression à minimiser contenant la somme de deux termes, $-logP_v(I|M)$ représentant la mesure de dissimilarité D de l'équation 2.26 et $-logP_P(M)$ pour l'énergie de complexité C de la même équation.

L'approche basée sur les **Champs de Markov** introduite par Geman et Geman [GEMAN84, GEMAN87, GEMAN90], consistant à modéliser une partition de l'image comme une réalisation d'un champ de Markov [LI01] fait également partie du cadre bayésien.

2.5.2.c Modélisation par codage minimal

A l'instar de la théorie bayésienne, les approches dites d'inférence par codage minimal se placent dans le cadre de l'inférence statistique de modèle mais proposent d'aborder le problème sous l'angle de la théorie de l'information. La théorie **MDL** (Minimum Description Length ou Longueur Minimale de Description) proposée par Rissanen [RISSAN81] est la plus connue dans le domaine du traitement d'image. Elle s'inspire de la notion de complexité de Kolmogorov [WALLAC99], qui considère que le meilleur modèle parmi un ensemble de modèles possibles est toujours celui permettant la représentation la plus compacte des données.

Le principe va donc consister à considérer un ensemble d'observations I ainsi que des modéles $\mathcal{M}$ permettant de les représenter. Le but étant ensuite d'exprimer de façon minimale I, en sélectionnant un modèle $M \in \mathcal{M}$ tel que la somme des coûts de codage de M et de I sachant M soit minimale. L'unité utilisée pour décrire le coût de codage (ou la complexité de la descritption) est le *bit*.

Etant donnée une image I et un modèle M décrivant cette image, une descritption complète (sans aucune perte) de I est donnée par :

$$\mathcal{L}_{1,2}(I < M >) = \mathcal{L}_1(M) + \mathcal{L}_2(I|M) \tag{2.34}$$

avec $\mathcal{L}_{1,2}(I < M >)$ la longueur de description (MDL) de I, $\mathcal{L}_1(M)$ représentant le coût de codage de I par le modèle M et $\mathcal{L}_2(I|M)$ décrivant les différences entre I et M ou le coût de codage de I sachant M.

On peut noter la similarité entre l'approche MDL et l'approche Bayesienne (Section 2.5.2.b). En effet, en théorie de l'information, les quantités $-logP_P(M)$ et $-logP_v(I|M)$ peuvent respectivement s'interpréter comme le coût de codage de M et le coût de codage de I sachant M.

2.5.3 Coupes optimales

Comme nous l'avons vu dans la Section 2.5.2, trouver une partition P minimisant une énergie E revient généralement à la résolution d'un problème difficile. Si maintenant, nous disposons d'une hiérarchie H et que nous ne nous intéressons qu'aux partitions pouvant être formées à partir d'éléments inclus dans H, c'est à dire à des coupes de H, nous allons voir que ce problème peut se résoudre plus facilement.

Soit H une hiérarchie et E_λ l'énergie affine séparable (Déf. 65) suivante :

$$E_\lambda(P) = \sum_{R_i \in P} \lambda C(R_i) + D(R_i) \tag{2.35}$$

avec C une énergie de régularisation et D une énergie d'attache aux données. Par commodité, nous noterons ce type d'énergie $E_\lambda = (C, D)$.

Pour tout $\lambda \in \mathbb{R}$, nous appelons **λ-coupe**, la coupe minimisant E_λ dans H et la notons $C^*_\lambda(H)$. La famille des λ-coupes de H est notée $C^*(H) = \{C^*_\lambda(H)\}_{\lambda \in \mathbb{R}^+}$. Une région $x \in H$ appartenant à la coupe $C^*_\lambda(H)$ est dite λ-optimale.

Définition 66 Énergie sous-additive

*Soit $\mathcal{R}(I)$, l'ensemble des régions connexes pouvant être définies sur une image I, une énergie $E : \mathcal{R}(I) \to \mathbb{R}^+$ peut être qualifiée de **sous-additive** si elle vérifie :*

$$\forall (x, x') \in \mathcal{R}(I) \times \mathcal{R}(I) \quad x \cap x' \neq \emptyset \quad : \quad E(x \cup x') \leq E(x) + E(x') \tag{2.36}$$

Notons que la sous-additivité d'une énergie de régions est équivalente à la décroissance de l'énergie séparable associée, ce qui conduit à la proposition suivante :

Proposition 1 *Une énérgie de partitions séparable $E : \mathbb{P}(X) \to \mathbb{R}^+$ est sous additive si et seulement si*

$$\forall (P, P') \in \mathbb{P}(X) \times \mathbb{P}(X) \quad P \geq P' \quad : \quad E(P) = \sum_{R \in P} E(R) \leq E(P') = \sum_{R' \in P'} E(R') \tag{2.37}$$

Ceci nous amène à un théorème central dans la théorie ensemble-échelle proposée par Guigues [GUIGUE06] :

Théorème 3 Coupes minimales causales

Soit H une hiérarchie et $E_\lambda = (C, D)$ une énergie séparable sur un ensemble X.

Si l'énergie C est sous additive alors l'ensemble $\{C^*_\lambda(H)\}_{\lambda \in \mathbb{R}^+}$ est causal

$$\forall (\lambda, \lambda') \in \mathbb{R}^{+2} \quad \lambda \geq \lambda' \quad : \quad C^*_\lambda(H) \geq C^*_{\lambda'}(H) \tag{2.38}$$

Notons que la sous-additivité du terme de régularisation, dans les énergies utilisées pour la segmentation d'images, se conçoit assez naturellement. Par exemple, dans le cas du codage d'un ensemble de points par leur moyenne, le stockage de x et x' impose de stocker deux moyennes alors que le codage de $x \cup x'$ permet de n'en stocker qu'une seule. De même, le terme de régularisation, employé par Mumford et Shah [MUMFOR89] défini dans la Section 2.5.2.a, prenant en compte la longueur totale des frontières est sous-additif. Un résultat équivalent pourrait également être obtenu si aucune condition n'était posée sur le terme C mais que le terme d'attache aux données D était contraint à être *sur-additif*, c'est à dire à vérifier :

$$\forall (x, x') \in H \quad D(x \cup x') \geq D(x) + D(x')$$

Le théorème 3 peut s'interpréter comme la monotonie des coupes minimales causales dans une hiérarchie. Ce théorème permet également de calculer efficacement un coupe optimale dans H pour tout paramètre λ

Pour tout $x \in H$ la coupe minimale $C^*(x)$ de la hiérarchie partielle $H(x)$ (voir Déf.47) peut se définir par :

$$C^*(x) = \underset{C \in \mathbb{C}ut(H(x))}{\operatorname{argmin}} E(C) = \underset{C \in \mathbb{C}ut(H(x))}{\operatorname{argmin}} \sum_{x' \in C} E(x') \tag{2.39}$$

Les énergies des coupes minimales des hiérarchies partielles $E(C^*(x))$ (que l'on notera $E^*(x)$ par souci de simplification) peuvent alors être calculées, par un algorithme bottom-up, grâce à la relation suivante :

$$\forall x \in H \quad E^*_\lambda(H(x)) = min\Big\{E_\lambda(x), \sum_{x' \in F(x)} E^*_\lambda(H(x'))\Big\} \tag{2.40}$$

La relation 2.40 traduit simplement le fait que l'énergie séparable $E^*_\lambda(H(x))$ qui sera attribuée à une région x, correspondra au minimum de l'énergie calculée sur cette région et de la somme des énergies des cuts optimaux de ces fils $F(x)$. Le cut optimal de la hiérarchie $C^*_\lambda(H)$ se calcule donc de façon ascendante en partant de la base de la hiérarchie et en calculant récursivement les énergies de cuts optimaux définies par l'équation 2.40. La compléxité de ce calcul est linéaire par rapport au nombre de régions dans la hiérarchie.

Considérons par exemple une région R de niveau 2 dans la hiérarchie (donc telle que chacun de ses fils soit une feuille, voir Figure 2.38(a)). L'énergie de la coupe optimale de chacun des fils $\{S_1, \ldots, S_n\}$ de R est alors réduite à l'énergie de la région considérée. L'énergie de la coupe optimale de R sera donc égale au min de $E_\lambda(R)$ et $\sum_{i=1}^{n} E_\lambda(S_i)$. Ce calcul effectué pour chaque sommet de niveau 2, peut se propager au niveau 3 jusqu'au sommet.

Le théorème 3 nous permet de calculer pour chaque région de la hiérarchie son intervalle de persistance $\chi(x) = [\lambda^+(x), \lambda^-(x)[$. Récrivons la relation de l'équation 2.40 comme une relation fonctionnelle, dans laquelle l'énergie de chaque région est fonction du paramètre d'échelle λ :

$$\forall x \in H \quad E^*_x(\lambda) = min\Big\{E_x(\lambda), \sum_{x' \in F(x)} E^*_{x'}(\lambda)\Big\} \tag{2.41}$$

L'intervalle de persistance d'un sommet x, correspond par définition à l'ensemble des valeurs de λ pour lesquelles $E^*_x(\lambda) = E_x(\lambda)$. Ces intervalles nous sont donnés par le théorème suivant du à Guigues [GUIGUE06] :

Proposition 2 *Soit H une hiérarchie et $E_\lambda = (C, D)$ une énergie multi-échelle affine séparable avec C sous-additive, alors pour tout $x \in H$:*

i La fonction E^*_x est affine par morceau, non-décroissante, continue et concave.

ii L'échelle d'apparition $\lambda^+(x)$ de $x \in H$ est l'unique solution de $E_x(\lambda) = \sum_{x' \in F(x)} E^*_{x'}(\lambda)$

iii $\lambda^-(x)$, $x \in H$ est donné par $\lambda^-(x) = min\{\lambda^+(p) | p \in H, x \subset p\}$.

Le calcul des intervalles de persistance est une généralisation de l'algorithme précédent. On effectue tout d'abord une étape ascendante similaire à celle de l'algorithme précédent qui calcule pour chaque sommet l'échelle d'apparition $\lambda^+(x)$ grâce à la proposition 2(ii). Une fois toutes les échelles d'apparitions calculées les échelles de disparitions se déduisent de celles-ci en utilisant la proposition 2(iii).

Pour illustrer ce calcul des échelles d'apparition de régions, considérons à nouveau la hiérarchie partielle $H(R)$ représentée sur la Figure 2.38(a). Comme R appartient au

second niveau, la hiérarchie partielle $H(R)$ admet seulement deux coupes distinctes. Celle qui code la partition P_1, composée des fils de R, dont l'énergie est égale à la somme :

$$E_\lambda(P_1) = \sum_{i=1}^{n} D(S_i) + \lambda \sum_{i=1}^{n} C(R_i)$$

et celle qui code la partition P_2, composée de l'unique région R. L'énergie de P_2 sera alors donnée par

$$E_\lambda(P_2) = D(R) + \lambda C(R)$$

Comme l'énergie C est sous-additive nous avons :

$$\sum_{i=1}^{n} C(R_i) > C(R)$$

En utilisant la représentation linéaire de $E_\lambda(P_1)$ et $E_\lambda(P_2)$ par rapport à λ, si l'énergie D est sur-additive, c'est à dire si :

$$\sum_{i=1}^{n} D(S_i) < D(R)$$

la droite représentant l'énergie de $E_\lambda(P_1) = \sum_{i=1}^{n} D(S_i) + \lambda \sum_{i=1}^{n} C(R_i)$ se trouve en dessous de celle représentant $E_\lambda(P_2) = D(R) + \lambda C(R)$ jusqu'à ce que λ atteigne la valeur $\lambda^+(R)$ pour laquelle les deux lignes représentant les énergies s'intersectent (Figure 2.38(b)). Dans le cas où

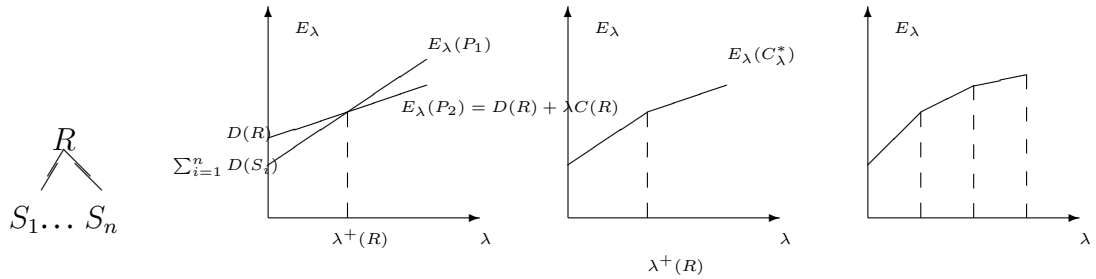
$$\sum_{i=1}^{n} D(S_i) \geq D(R)$$

l'énergie $E_\lambda(P_2)$ est toujours supérieure ou égale à $E_\lambda(P_1)$ et $\lambda^+(R)$ est alors égal à 0. Dans tous les cas de figure, la partition P_1 est associée à une énergie inférieure à celle de P_2 pour $\lambda = 0$ jusqu'à $\lambda = \lambda^+(R)$. Au delà de cette valeur P_2 est alors associée à l'énergie la plus faible. Si nous raisonnons maintenant en terme de coupe optimale, la partition P_1 correspond donc à la coupe optimale de la hiérarchie partielle $H(R)$ pour des valeurs de λ inférieures à $\lambda^+(R)$ alors que la partition P_2 correspond à la coupe optimale pour les valeurs de λ supérieures à $\lambda^+(R)$ ((Figure. 2.38(c)).

Après avoir effectué les deux traversées de la hiérarchie initiale H, l'une montante pour obtenir les λ^+ et l'autre descendante pour obtenir les λ^-, il peut apparaître des régions qui n'appartiendront à aucune coupe. Ce type de région intervient quand son échelle d'apparition est inférieure à celle de disparition. Ces régions disparaissent alors avant même d'être apparues et sont qualifiées de **non persistantes**

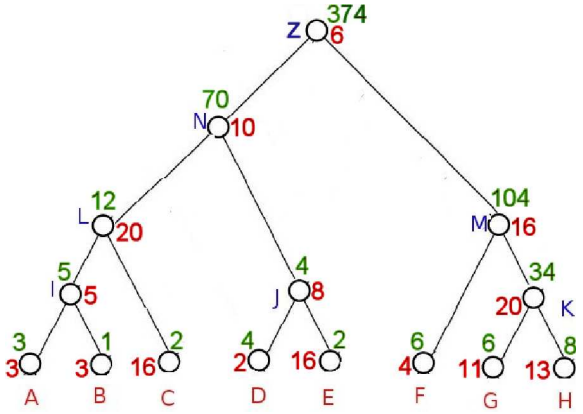
$$\exists x' \in H \quad | \quad x \subset x' \quad et \quad \lambda^+(x') \leq \lambda^-(x) \tag{2.42}$$

Les régions non persistantes n'ayant aucun intérêt, du point de vue des minima de la fonction d'énergie, il est naturel de les supprimer de la hiérarchie en les fusionnant à leurs pères. Une telle hiérarchie, notée H^*, privée des régions non persistantes est alors qualifiée de hiérarchie persistante pour E_λ.



(a) H(R) (b) $E_\lambda(P_1), E_\lambda(P_2)$ (c) $E_\lambda(C^*_\lambda(H(R)))$ (d) $E_\lambda(C^*_\lambda(H))$

FIG. 2.38 – (a) un nœud R appartenant à la hiérarchie dont les fils $\{S_1, \dots, S_n\}$ correspondent aux régions initiales. (b) Energies des partitions associées à R et à $\{S_1, \dots, S_n\}$ représentées comme des fonctions de λ. (c) Energie de la coupe optimale par rapport à $H(R)$ (a). (d) exemple de fonction concave, affine par morceau codant l'énergie des coupes optimales dans la hiérarchie H.



(a) énergies C et D dans H

FIG. 2.39 – Arbre représentant une hiérarchie H. Pour chaque sommet les énergies C sont écrites en rouge (à côté) et les énergies D en vert (au dessus)

Définition 67 *Hiérarchie persistante*

Si H est une hiérarchie et E_λ une énergie multi-échelle, alors le sous-ensemble de H

$$H^* = \{x \in H \; \chi^*(x) \neq \emptyset\}$$

*est une hiérarchie appelée **hiérarchie persistante** (pour E_λ).*

2.5.3.a Exemple de construction d'une hiérarchie de coupes optimales

Soit H une hiérarchie construite à partir de fusions successives effectuées sur une partition initiale P. La figure 2.39 illustre une telle hiérarchie, la partition initiale correspond à la première partition de la figure 2.41. A chacun des sommets de cette hiérarchie, représentant une région de la partition P, sont affectées une énergie d'attache aux données D (en vert et au dessus des nœuds sur la Figure 2.39) ainsi qu'une énergie de régularisation

(en rouge et sur le côté des nœuds). Il est alors possible de calculer pour chaque sommet, son échelle d'apparition λ^+ en utilisant l'équation 2.41.

Détaillons par exemple les calculs de $\lambda^+(I)$, λ^+(L) et λ^+(N) :

Nous allons donc calculer l'énergie optimale du sommet I

$$E_I^* = \min(E_I, \sum_{s \in F(L)} E_s^*) = \min(E_I, E_A^* + E_B^*)$$

A et B étant des feuilles de la hiérarchie, elles n'ont qu'une seule énergie et l'on a :

$$\left\{ \begin{array}{rcrcl} E_A^* & = & E_A & = & 3\lambda + 3 \\ E_B^* & = & E_B & = & 3\lambda + 1 \end{array} \right.$$

On a donc :

$$E_I^* = \min(E_I, E_A^* + E_B^*) = \min(5\lambda + 5, 6\lambda + 4)$$

Les deux droites se coupent en $\lambda = 1$ et on a :

$$E_I^* = \left\{ \begin{array}{ll} 6\lambda + 4 & \text{Si } \lambda \leq 1 \\ 5\lambda + 5 & \text{Sinon} \end{array} \right.$$

Étudions à présent l'énergie du noeud L :

$$E_L^* = \min(E_L, \sum_{s \in F(L)} E_s^*) = \min(E_L, E_C^* + E_I^*)$$

Le noeud C étant une feuille nous avons $E_C^* = E_C = 16\lambda + 2$ et :

$$E_C^* + E_I^* = \left\{ \begin{array}{ll} 22\lambda + 6 & \text{Si } \lambda \leq 1 \\ 21\lambda + 7 & \text{Sinon} \end{array} \right.$$

en comparant avec l'énergie de $E_L = 20\lambda + 12$, nous obtenons :

- si $\lambda \leq 1$:

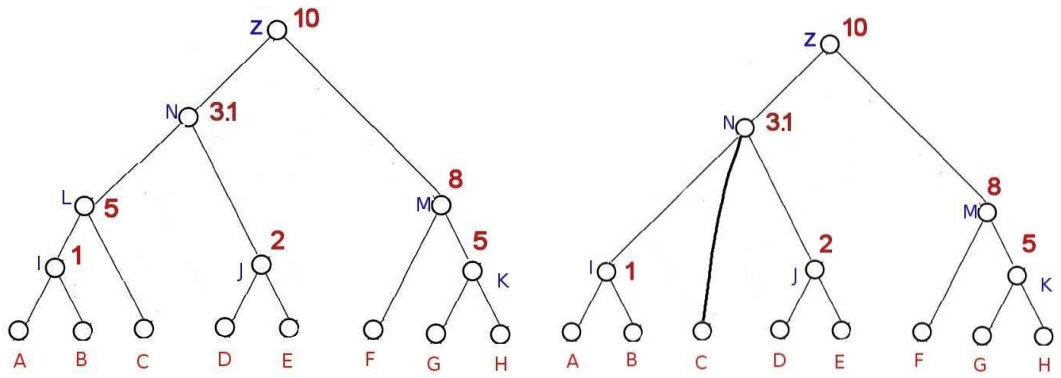
$$\left. \begin{array}{l} \sum_{s \in F(L)} E_s^* = E_I^* + E_C^* = 22\lambda + 6 \\ E_L = 20\lambda + 12 \end{array} \right\} \lambda_L^+ = 3$$

  cette solution est rejetée du fait que λ ne peut être à la fois inférieur à 1 et égal à 3.
- si $\lambda \geq 1$:

$$\left. \begin{array}{l} \sum_{s \in F(L)} E_s^* = E_I^* + E_C^* = 21\lambda + 7 \\ E_L = 20\lambda + 12 \end{array} \right\} \lambda_L^+ = 5$$

L'échelle d'apparition du sommet L sera donc $\lambda_L^+ = 5$ et on a :

$$E_L^* = \left\{ \begin{array}{rcll} E_A + E_B + E_C & = & 22\lambda + 6 & \text{Si } 0 \leq \lambda \leq 1 \\ E_I + E_C & = & 21\lambda + 7 & \text{Si } 1 \leq \lambda \leq 5 \\ E_L & = & 20\lambda + 12 & \text{Si } \lambda \geq 5 \end{array} \right.$$



(a) λ^+ pour chacun des sommets de H (b) hiérarchie persitante H^*

FIG. 2.40 – arbre représentant une hiérarchie H (a) avec l'échelle d'apparition de chaque sommet. (c) La hiérarchie persistante H^* après élagage du sommet non persistant L.

Étudions à présent le sommet N. On a :

$$E^*_N = \min(E_N, E^*_L + E^*_J)$$

On montre facilement que :

$$E^*_J = \begin{cases} E_D + E_E = 9\lambda + 2 & \text{Si } 0 \leq \lambda \leq 2 \\ E_J = 8\lambda + 4 & \text{Si } \lambda \geq 2 \end{cases}$$

On a donc, en intersectant les intervalles :

$$E^*_L + E^*_J = \begin{cases} E_A + E_B + E_C + E_D + E_E = 31\lambda + 8 & \text{Si } 0 \leq \lambda \leq 1 \\ E_I + E_C + E_D + E_E = 30\lambda + 9 & \text{Si } 1 \leq \lambda \leq 2 \\ E_I + E_C + E_J = 29\lambda + 11 & \text{Si } 2 \leq \lambda \leq 5 \\ E_L + E_J = 28\lambda + 6 & \text{Si } \lambda \geq 5 \end{cases}$$

On constate que la polyline $E^*_L + E^*_J$ intersecte la droite $E_N = 10\lambda + 70$ en $\lambda = 3.1$. Cela signifie que le noeud L n'intervient pas dans les coupes optimales de N et qu'il devra être rejeté de la hiérarchie persistante. L'énergie du noeud N est donc :

$$E^*_N = \begin{cases} E_A + E_B + E_C + E_D + E_E = 31\lambda + 8 & \text{Si } 0 \leq \lambda \leq 1 \\ E_I + E_C + E_D + E_E = 30\lambda + 9 & \text{Si } 1 \leq \lambda \leq 2 \\ E_I + E_C + E_J = 29\lambda + 11 & \text{Si } 2 \leq \lambda \leq 3.1 \\ E_N = 10\lambda + 70 & \text{Si } \lambda \geq 3.1 \end{cases}$$

Une fois l'ensemble des échelles d'apparition λ^+ calculées (Figure 2.40(a)), il reste à effectuer une étape d'élagage sur les sommets non persistants de la hiérarchie. Dans cet exemple, le sommet L doit être rattaché à son père N. La hiérarchie persistante H^* obtenue est représentée sur la Figure 2.40(b). La création du sommet non persistant L peut être due à une limitation de la stratégie de découpe (Section 2.6) qui a imposé de «passer» par le sommet L avant de créer le sommet N. Il est assez remarquable que le calcul des coupes optimales nous permette de rejeter ce sommet non pertinent et donc de corriger les résultats induits par notre stratégie de construction de la pyramide.

La donnée du couple (H^*, λ^+) nous fournit ainsi une représentation ensemble échelle des coupes optimales de H, une hiérarchie indicée dont la famille des coupes naturelles est

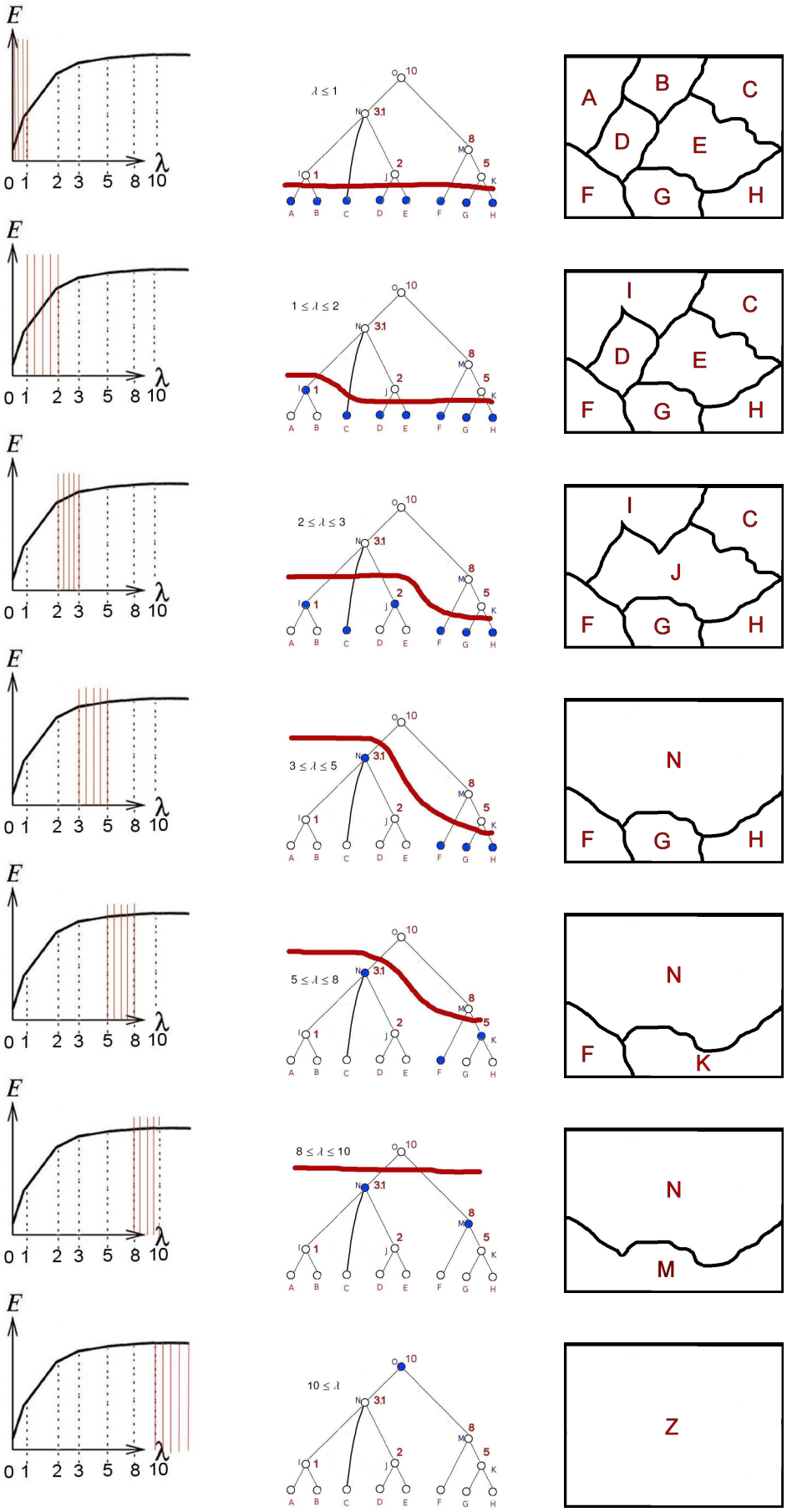


FIG. 2.41 – Représentation des différentes coupes optimales dans la hiérarchie H^* de la Figure. 2.40(b). La 1ère colonne représente la fonction affine par morceaux avec les différents intervalles de persistance, la 2nde présente les coupes optimales pour les différentes échelles d'apparition. La 3ème présente la partition correspondant à chaque intervalle.

la famille de coupes optimales de H pour une énergie E_λ causale donnée. La Figure 2.40 illustre ces propos en présentant cette famille de coupes optimales avec les partitions qui lui sont associées.

2.6 Construction de hiérarchies

Nous avons vu dans la section 2.4.1 un codage des relations hiérarchiques à l'aide d'un arbre représentant les processus de fusion. Inversement, la section 2.4.3 a présenté un ensemble de modèles hiérarchiques basés sur un codage de la hiérarchie à l'aide d'une suite de graphes. Notons que les deux types de codages ne sont pas incompatibles et sont même souvent utilisés conjointement. En effet, l'utilisation d'au moins un graphe codant les relation d'adjacences est nécessaire pour construire une hiérarchie basée «arbre». Inversement, les relations inter-niveaux d'une pyramide basée «graphe» induisent des relations père/fils que l'on peut représenter à l'aide d'un arbre.

Dans cette section nous allons nous focaliser sur les différentes heuristiques de construction des pyramides irrégulières. Ces heuristiques considèrent généralement la pyramide comme une pile de graphe réduit $P = (G_0, \ldots, G_n)$. Les heuristiques présentées dans les sections suivantes permettent de construire un graphe G_{l+1} à partir du graphe courant G_l.

2.6.1 Décimation stochastique

Meer [MEER89A] définit l'opération de réduction d'un graphe comme un processus stochastique. Le passage du graphe $G_l = (V_l, E_l)$ de niveau l au graphe $G_{l+1} = (V_{l+1}, E_{l+1})$ du niveau suivant s'effectue en 3 étapes :

1. sélectionner par un processus de décimation stochastique un sous-ensemble de sommets $V_{l+1} \subset V_l$ qui seront les sommets survivants.
2. définir une partition de V_l en établissant une liaison entre chacun des sommets non survivants et ceux qui vont survivre au niveau suivant.
3. définir l'ensemble d'arêtes E_{l+1} définissant les relations d'adjacence entre les sommets de G_{l+1}.

Afin d'obtenir un taux de décimation constant Meer impose sur l'ensemble de sommets survivants les deux contraintes de stabilité suivantes :

1. Stabilité externe :

$$\forall v \in V_l - V_{l+1}\ \exists v' \in V_{l+1} :\ (v, v') \in E_l. \tag{2.43}$$

2. Stabilité interne :

$$\forall (v, v') \in V_{l+1}^2 :\ (v, v') \notin E_l. \tag{2.44}$$

La contrainte (2.43) impose à chaque sommet non survivant d'être adjacent à au moins un sommet survivant. La contrainte (2.44) interdit à deux sommets adjacents de survivre simultanément. Le sous ensemble des sommets d'un graphe qui possède conjointement ces deux propriétés de stabilité est appelé un *noyau* [ROY69].

Soit $G_O = (V_0, E_0)$ le graphe initial défini sur une image I. Meer définit un noyau sur l'ensemble des sommets du graphe à l'aide d'un processus stochastique. Une valeur réelle aléatoire $\chi_i \in [0, 1]$ (suivant une distribution uniforme) est associée à chaque sommet du graphe. Les sommets survivants sont ceux qui réalisent un maximum local de la valeur de χ_i (i.e la valeur maximale dans le sous ensemble constitué par un sommet et tous ses voisins). Comme deux sommets adjacents ne peuvent correspondre tous deux à un maximum local, deux sommets voisins ne peuvent appartenir simultanément à l'ensemble des sommets survivants. Ceci vérifie la condition de stabilité interne (condition 2.44).

Nous pouvons noter que cette méthode prend en compte implicitement le degré d'un sommet. En effet plus le degré d'un sommet est élevé, plus la probabilité que la valeur de sa variable aléatoire soit supérieur à tous les tirages de ses voisins est faible. Les sommets présentant une faible arité sont donc favorisés par cette approche. Une seule itération de ce processus peut néanmoins laisser un sommet, ne correspondant pas à un maximum local, avec seulement des sommets non survivant dans son voisinage. Il convient donc d'itérer ce processus jusqu'à ce que les deux conditions (2.43 et 2.44) soient vérifiées simultanement.

Pour cela 3 variables χ_i, p_i et q_i sont attachées à chaque sommet $v_i \in V_l$ du graphe courant G_l. La variable χ_i code le tirage de la variable aléatoire pour le sommet v_i. Les variables p_i et q_i sont deux booléens dont la valeur code les états suivants :

- si p_i est vrai, le sommet v_i est considéré comme un sommet survivant ;
- si q_i est vrai, v_i peut devenir un sommet survivant lors des itérations futures.

Soient $(p_i^{(k)})_{k\in\{1,\ldots,n\}}$ et $(q_i^{(k)})_{k\in\{1,\ldots,n\}}$ les différentes valeurs de p_i et q_i au cours des n itérations de l'algorithme. L'initialisation de ces variables sera effectuée de la façon suivante lors de la première itération :

$$\begin{array}{rcl} p_i^{(1)} & = & x_i = max_{j\in V(v_i)}\{x_j\} \\ q_i^{(1)} & = & \bigwedge_{j\in V(v_i)} \overline{p_j}^{(1)} \end{array} \tag{2.45}$$

avec $V(v_i)$ représentant l'ensemble des sommets adjacents à v_i et $\bigwedge$ l'opérateur logique "et". Par convention, le sommet v_i appartient à son voisinage $V(v_i)$. Un sommet sera donc survivant s'il est un maximum local de la variable aléatoire.

Pour les étapes suivantes les mises à jour des prédicats seront effectuées de la façon suivante :

$$\begin{array}{rcl} p_i^{(k+1)} & = & p_i^{(k)} \vee \ (q_i^{(k)} \wedge \ x_i = max_{j\in V(va_i)}\{q_j^{(k)} x_j\}) \\ q_i^{(k+1)} & = & \bigwedge_{j\in V(v_i)} \overline{p_j}^{(k+1)} \end{array} \tag{2.46}$$

Avec le symbole $\vee$ représentant le "ou" logique. Un sommet désigné survivant à l'étape k le restera à l'étape suivante ($p_i^{(k+1)} = p_i^{(k)} \vee \ldots$). Un sommet non survivant pourra le devenir s'il réalise un maximum local de sa variable aléatoire sur l'ensemble des sommets de son voisinage pouvant devenir survivant (avec le prédicat q égal à 1). Enfin un sommet qui n'est pas désigné comme survivant mais dont aucun des voisins ne survit restera un candidat à la survie. Ce processus est itéré tant que l'ensemble de sommets sélectionnés ne forme pas un noyau. Notons que ce type de processus s'effectue de manière purement locale. En effet, à chaque itération, un sommet calcule son état uniquement en fonction de son état précédent et de celui de ses voisins. Un tel processus peut donc aisément être implémenté sur une machine parallèle.

L'établissement des liaisons parents-enfants s'effectue simplement en attachant chaque sommet non survivant au sommet adjacent survivant, dont le tirage de la variable aléatoire est maximum. La dernière étape consiste à connecter les sommets survivants dans le graphe G_{l+1}. Deux pères seront donc reliés par une arête si leurs fenêtres de réductions dans le graphe G_l sont adjacentes par au moins un sommet.

Ce type de processus a été utilisé dans le cadre de la génération de motifs aléatoires [MEER89B] ainsi que pour l'étude de la stabilité des algorithmes pyramidaux [MEER88]. Cette approche purement aléatoire a été tout d'abord adaptée par Montanvert [MONTAN91] à l'analyse de composantes connexes, en restreignant le processus de décimation à un ensemble de sous-graphes du graphe initial. Deux sommets survivants peuvent alors être adjacents dans le graphe à décimer s'ils n'appartiennent pas au même sous-graphe. Jolion [JOLI92] propose d'améliorer la flexibilité du processus de décimation en utilisant les maxima locaux d'un opérateur d'intérêt plutôt que les maxima locaux des tirages de la variable aléatoire. Il propose ainsi un critère tendant à sélectionner les sommets survivants situés dans les régions homogènes, ouvrant ainsi la porte à de nombreuses applications pour la segmentation en régions homogènes [JOLION93, BEN YA95, CHIARE96].

2.6.2 Décimation guidée par les données

Jolion [JOLI01] propose un nouveau modèle de pyramide adaptative dans lequel le choix des sommets survivants n'est plus fait de manière stochastique mais est dépendant du contenu de l'image. L'idée de base de ce modèle consiste à appliquer un processus de décimation ne s'effectuant qu'en une seule itération à chaque niveau de la pyramide. Deux variables booléennes p_i et q_i, définies de la même façon que dans la Section 2.6.1, sont attachées aux sommets v_i du graphe. Elles conservent la même signification mais sont désormais globales à l'ensemble de la pyramide. Elle sont dans un premier temps initialisées à vrai au niveau de base $G_0 = (V_0, E_0)$ de la pyramide

$$p_i^{(0)} = q_i^{(0)} = vrai, \quad \forall v_i \in V_0$$

et leur mise à jour entre chaque niveau de la pyramide est réalisée par les équations suivantes :

$$\begin{aligned} p_i^{(k+1)} &= \left((p_i^{(k)} \vee q_i^{(k)}) \wedge x_i = max_{j \in V_k(v_i)} \{ q_j^{(k)} x_j \} \right) \\ q_i^{(k+1)} &= \bigwedge_{j \in V_k(v_i)} \overline{p_j}^{(k+1)} \wedge (V_k(v_i) \neq \{v_i\}). \end{aligned}$$

Un sommet est donc déclaré survivant s'il était survivant ou candidat au niveau précédent et s'il réalise un maximum de la variable aléatoire dans le graphe G_k. Il est déclaré candidat s'il n'est ni adjacent à un sommet survivant ni isolé. Les sommets du graphes sont ainsi répartis en trois classes disjointes : les sommets survivants, candidats et non candidats. Les sommets non candidats, nécessairement adjacents à un des sommets survivants, sont réduits alors que les sommets survivants font partie de l'ensemble de sommets du graphe réduit. Les décisions nécessaires pour classer les sommets candidats seront désormais prises à des niveaux plus élevés de la pyramide, l'ensemble des sommets V_{k+1} de G_{k+1} :

$$V_{k+1} = \{v_i \in V_k \mid p_i^{(k+1)} \vee q_i^{(k+1)}\}$$

sera alors composé des sommets survivants et candidats. L'établissement des arêtes E_{k+1} entre les sommets de E_{k+1} est effectué par le même processus que dans la Section 2.6.1.

Cette approche à été conçue avec comme motivation principale la prise en compte de deux types de régions lors du processus : les régions les plus homogènes ou celles présentant un intérêt particulier vis à vis du critère de décimation. En effet dans ce type de processus asynchrone, les régions présentant un faible intérêt ont tendance à fusionner en premier. Ce processus permet ainsi de détecter ces régions de façon précoce afin de conserver les régions les plus pertinentes (au sens du critère de «désintérêt» x_i utilisé), dans les niveaux plus élevés de la pyramide.

2.6.3 Décimation par couplage maximal

Comme nous l'avons vu dans la Section. 2.6.1 la méthode de décimation proposée par Meer [MEER89A] et Montanvert [MONTAN91] est basée sur la définition d'un noyau. De ce fait la contrainte de stabilité interne (équation 2.44) va influer fortement sur le nombre de sommets survivants. En effet un sommet est désigné survivant s'il réalise un maximum des tirages d'une variable aléatoire sur son voisinage. Plus le degré d'un sommet est élevé, plus la probabilité que la valeur de sa variable aléatoire soit supérieure à tous les tirages de ses voisins est faible. La probabilité *a priori* qu'un sommet survive est donc directement liée à la taille de son voisinage et la relation d'adjacence entre les sommets influe donc directement sur la hauteur de la pyramide et le nombre d'itérations nécessaire à la construction de chaque niveau de la pyramide.

Haxhimusa et al. [HAXHIM02B] ont montré que l'arité moyenne des sommets a tendance à augmenter dans la pyramide ce qui a pour conséquences une diminution du facteur de décimation et une augmentation de la taille de la pyramide. Ceci peut s'avérer gênant puisque l'objectif implicite des méthodes à noyau est d'avoir une hauteur ayant un ordre de grandeur proche du log de la taille du graphe initial. Pour pallier cet inconvénient Haxhimusa propose un processus de décimation, indépendant de l'arité des sommets, basé sur la définition d'un couplage maximal [ROY69].

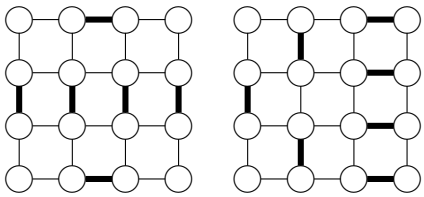
Définition 68 Couplage maximal

Étant donné un graphe $G = (V, E)$, un ensemble $C \subset E$ est appelé un couplage si aucune des arêtes de C n'est adjacente au même sommet.

- *un couplage est dit maximal si on ne peut y ajouter d'arête sans perdre la propriété de couplage ;*
- *un couplage est dit maximum s'il comporte un nombre maximum d'arêtes ;*
- *un couplage est dit parfait si tout sommet est incident à une arête de C. Un couplage parfait est maximum.*

Un sommet incident aux arêtes d'un couplage est dit saturé sinon il est dit insaturé.

La méthode proposée par Haxhimusa et al. définit une forêt K du graphe initial G telle que chaque arbre est composé *d'au moins* deux sommets et chaque sommet de G

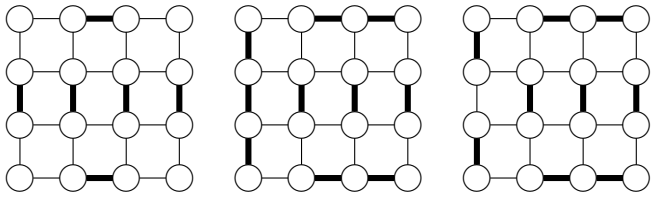


(a) 6 arêtes (b) 7 arêtes

FIG. 2.42 – Deux couplages maximaux comportant un nombre différent d'arêtes. Les arêtes faisant partie du couplage sont représentées en gras.

appartient à *exactement* un arbre de la forêt. La construction d'une telle forêt suit donc les étapes suivantes :

1. construire un couplage maximal C sur G ;
2. Supprimer les sommets insaturés en ajoutant de nouvelles arêtes. Le nouvel ensemble noté C^+ ne constitue plus un couplage. En revanche si le couplage est maximum, l'ensemble C^+ forme un recouvrement minimum de G [BERGE83] ;
3. supprimer des arêtes dans C^+ de façon à construire une forêt K composée d'arbres de profondeur 1.



(a) C (b) C^+ (c) K

FIG. 2.43 – Les trois étapes du processus de décimation de Haxhimusa [HAXHIM02B]

L'ensemble K obtenu forme ainsi une forêt du graphe G exclusivement composée d'arbres de profondeur 1. Une racine est désignée pour chaque arbre de K, ce qui induit une sélection naturelle de l'ensemble des sommets survivants et non survivants. Pour chaque arbre la racine est désignée comme sommet survivant alors que les fils de celle-ci sont classifiés comme sommets non survivants. L'établissement des liaisons pères-fils ainsi que celles entre les sommets survivants est effectuée de façon similaire à celle définie dans la Section. 2.6.1.

Le principal avantage de ce processus de décimation tient au fait qu'il permet de maintenir, tout au long des niveaux de la pyramide, un facteur de décimation plus stable. Haxhimusa a montré par l'expérimentation que ce facteur de décimation reste en moyenne supérieur à 2 tout au long du processus de construction de la pyramide.

2.6.4 Décimation par la méthode d'escalade

Guigues [GUIGUE06] a récemment proposé une façon naturelle de construire une hiérarchie en s'appuyant sur un principe d'optimisation d'énergie. Le principe d'escalade

qu'il propose s'inscrit dans une méthodologie de segmentation multi-échelle basée sur l'optimisation d'une énergie multi-échelle E_λ (Section 2.5.3). Nous avons présenté dans la section 2.5.3 comment, pour une hiérarchie et une énergie E_λ donnée , il est possible de calculer et de représenter sa séquence complète de coupes optimales. Cette méthode d'optimisation étant générale, elle peut être appliquée sur tout type de hiérarchie construit par des méthodes de fusion ou de division récursives de l'image. Guigues propose d'adapter le processus de construction d'une hiérarchie afin de prendre en compte explicitement la minimisation de l'énergie E_λ.

Supposons donc que nous disposons d'une énergie multi-échelle $E_\lambda = (C, D)$ permettant d'attribuer une échelle d'apparition aux éléments d'une hiérarchie. Quand on réalise l'union d'un certain nombre de composantes hiérarchiques, le sommet correspondant à l'union de ces composantes va être doté d'une échelle d'apparition pour E_λ.

Définition 69 Escalade pure [GUIGUE06]

Si X est un ensemble et E_λ une énergie multi-échelles,
- *Pour toute pré-hiérarchie (Déf. 2.23) H sur X, une opération **d'ecalade pure** sur H pour E_λ consiste à réaliser l'union de sommets de ses composantes hiérarchiques réalisant la plus faible échelle d'apparition λ^+ pour E_λ.*
- *La **hiérarchie d'escalade pure** sur X pour $E_,$ est la hiérarchie obtenue en itérant l'opération d'escalade pure à partir de la pré-hiérarchie minimale $H = \{\{x\}\}_{x\in X}$ (la sur-partition absolue de X).*

Comme nous l'avons vu dans la Section 2.5.3, si E_λ est une énergie multi-échelle alors les coupes optimales d'une hiérarchie remontent en même temps que le paramètre d'échelle λ augmente. Disposant désormais d'une hiérarchie incomplète, le critère d'escalade pure va choisir de compléter la hiérarchie, par l'ensemble qui serait le premier atteint par la remontée des coupes optimales si nous augmentions progressivement la valeur du paramètre λ. Le critère d'escalade considère donc à chaque étape toutes les unions d'ensembles possibles et sélectionne celle faisant apparaître le plus petit paramètre λ^+, ce qui peut s'interpréter comme la plus faible régularisation. Nous pouvons noter que chacune des étapes d'une escalade pure est persistante. Les étapes non persistantes pour l'énergie, qui n'apparaîtraient jamais dans une coupe optimale, sont naturellement supprimées.

L'escalade pure n'est cependant pas possible à mettre en œuvre en pratique. En effet, si à une étape donnée, H est constituée de N composantes, il est nécessaire d'étudier les 2^N cas de figure possibles, ce qui n'est combinatoirement pas envisageable. Guigues propose donc de limiter l'espace de recherche à l'ensemble des fusions de paires de régions voisines, ce qui définit la notion d'**escalade binaire**.

Pour tout couple de régions $(R_i, R_j) \in X$ telles que R_i appartient au voisinage de R_j, noté $R_i \in \mathcal{V}(R_i)$, le facteur d'apparition $\lambda_{app}(R_i \cup R_j)$ de l'union des deux régions $R_i \cup R_j$ est calculé de la façon suivante :

Soient E_{R_i} et E_{R_j} les énergies des régions R_i et R_j

$$E_{R_i}(\lambda) = \lambda C_{R_i} + D_{R_i} \tag{2.47}$$

$$E_{R_j}(\lambda) = \lambda C_{R_j} + D_{R_j} \tag{2.48}$$

et $E_{R_i \cup R_j}$ l'énergie de la région $R_i \cup R_j$ résultant de la fusion de R_i et R_j

$$E_{R_i \cup R_j}(\lambda) = \lambda C_{R_i \cup R_j} + D_{R_i \cup R_j} \tag{2.49}$$

En sommant les équations 2.47 et 2.48 et en combinant avec l'équation 2.49 on obtient l'échelle d'apparition de la région $R_i \cup R_j$:

$$\lambda_{app}(R_i \cup R_j) = \frac{D_{R_i} + D_{R_j} - D_{R_i \cup R_j}}{C_{R_j} + C_{R_i} - C_{R_i \cup R_j}} \tag{2.50}$$

Nous fusionnerons alors le couple de régions (R, R') tel que $\lambda_{app}(R \cup R')$ soit minimum.

En itérant ce calcul pour chacune des pré-hiérarchies obtenues, nous aboutissons finalement à une hiérarchie d'escalade (H^*, λ^*) dont des propriétés importantes ont été démontrées par Guigues [GUIGUE06].

Sur l'exempe de la figure Figure 2.40, nous allons obtenir les fusions dans l'ordre suivant : $A \cup B$ puis $D \cup E$ suivi de $I \cup C$ et $G \cup H$ puis $J \cup L$ suivi de $F \cup K$ et enfin $M \cup N$. La création du nœud L est dans ce cas induite par la stratégie d'escalade binaire, même si comme le montre le calcul des coupes optimales (Section 2.5.3.a) ce choix n'est pas optimal.

Comme le montre l'équation 2.50 nous obtenons ainsi un algorithme totalement exempt de paramètre d'échelle et basé uniquement sur le couple (C, D) d'énergies antagonistes. L'escalade se présente alors comme une stratégie gloutonne maximisant la gain immédiat de co-énergie. Guigues approxime la stratégie d'escalade pure par une escalade binaire. Nous verrons dans la Section 3 que cette restriction peut être en partie levée pour parvenir à des énergies plus faibles.

CONTRIBUTION À LA CONSTRUCTION DE HIÉRARCHIES ROBUSTES

Sommaire

Le chapitre qui précède a présenté les différentes structures utiles pour construire et représenter des hiérarchies. Les Sections 2.5 et 2.6 en particulier ont présenté différentes approches permettant la construction de hiérarchies, basée sur différents critères énergétiques.

Les algorithmes de construction de hiérarchies de régions reposent généralement sur un processus itératif de regroupement de régions. Dans le domaine du traitement d'images les fusions de régions sont généralement effectuées entre des régions deux à deux adjacentes. Cette méthode de construction conduit à des hiérarchies strictement binaires, ne prenant pas en compte l'ensemble des fusions possibles, menant ainsi à des énergies sous optimales.

Nous allons présenter dans ce chapitre différentes heuristiques permettant de construire des hiérarchies de régions. Ces méthodes sont basées sur l'approche ensemble-échelle proposée par Guigues [GUIGUE03] (Section 2.6.4) et permettent d'obtenir un compromis entre le temps d'exécution et l'énergie des partitions obtenues.

Nous allons dans un premier temps (Section 3.1) proposer une normalisation des courbes $E_\lambda(C^*_\lambda(H))$ décrivant l'énergie des coupes d'une hiérarchie H en fonction du paramètre d'échelle λ. Cette normalisation nous permettra de comparer des courbes calculées sur des images différentes. Nous verrons dans un second temps (Section 3.2) une première approche descendante de calcul d'une hiérarchie. Cette approche n'ayant pas donnée les résultats escomptés nous proposons dans la Section 3.3 deux nouvelles heurstiques basées sur une approche ascendante. La première heuristique (Section 3.3.1) permet d'élargir l'espace de recherche en prenant en compte des hiérarchies n-aires, permettant à plusieurs régions de fusionner avec une région centrale à chaque étape de la construction. La deuxième heuristique, détaillée dans la Section 3.3.2, propose de paralléliser le processus de décimation pour améliorer le temps d'exécution aussi bien sur des machines parallèles que sur des machines séquentielles.

3.1 Normalisation d'énergie séparable

Nous allons dans cette section, nous intéresser aux propriétés des énergies séparables et plus particulièrement à leurs bornes supérieures et inférieures.

Considérons une énergie affine séparable de partitions définie pour toute partition P par :

$$E_\lambda(P) = D(P) + \lambda C(P)$$

avec D quelconque et C sous additive.

Comme nous l'avons vu dans la Section 2.5.3 , partant d'une hiérarchie H donnée, l'ensemble des coupes optimales $C^*_\lambda(H)$, défini pour cette hiérarchie, représente une segmentation multi-échelle non biaisée.

Si l'on désigne respectivement par C_I et D_I le terme de régularisation et l'attache aux données de la partition réduite à une seule région contenant l'image I on a, du fait de la sous additivité de C :

$$\forall P \in \mathbb{P}, \quad C(P) \geq C_I \tag{3.1}$$

L'énergie des coupes optimales $E_\lambda(C^*_\lambda(H))$ est une fonction concave linéaire par morceaux en fonction de λ (voir Proposition 2, Section 2.5.3). Chaque partie linéaire de $E_\lambda(C^*_\lambda(H))$ correspond à l'énergie d'une coupe optimale $C^*_\lambda(H)$, les coupes étant décroissantes en fonction de λ. Si nous supposons que la coupe optimale la plus grossière P_{max} n'est pas égale à la partition triviale réduite à une seule région nous obtenons :

$$\forall \lambda \geq \lambda_{max}, \quad E_\lambda(C^*_\lambda(H)) = D(P_{max}) + \lambda C(P_{max}) \leq D_I + \lambda C_I$$

où λ_{max} dénote l'échelle au delà de laquelle $C^*_\lambda(H) = P_{max}$. La seconde inégalité est due au fait que la partition triviale fait partie des partitions possibles et est jugée non optimale. Or l'équation 3.1 nous indique que $C(P_{max}) \leq C_I$. L'inégalité $E_\lambda(C^*_\lambda(H)) \leq D_I + \lambda C_I$ est donc impossible pour tout $\lambda \geq \lambda_{max}$ et ce quelles que soient $D(P_{max})$ et D_I. La partition optimale la plus grossière P_{max} est donc forcément égale à la partition triviale et on a :

$$E_\lambda(P_{max}) = D_I + \lambda C_I \tag{3.2}$$

La droite $E_\lambda(P_{max})$ peut se concevoir comme la tangente à la courbe $E_\lambda(C^*_\lambda(H))$ pour $\lambda \geq \lambda_{max}$. La fonction $E_\lambda(C^*_\lambda(H))$ étant concave elle est obligatoirement au dessous de sa tangente. L'énergie des coupes optimales $E_\lambda(C^*_\lambda(H))$ sera donc toujours bornée par l'énergie $E_\lambda(P_{max})$ associée à la partition la plus grossière (Fig. 3.1).

Soit P_0 la partition initiale correspondant soit à la grille des pixels soit à une sursegmentation initiale (ligne de partage des eaux par exemple). Dans ce cas les points $(0, E_0(P_0))$ et $(\lambda_{max}, E_{\lambda_{max}}(P_{max}))$ appartiennent tous deux à la courbe. L'énergie $E_\lambda(C^*_\lambda(H))$ étant concave, sa courbe représentative doit se situer au dessus de la ligne connectant ces deux points.

La ligne reliant les points $(0, 0)$ et $(\lambda_{max}, E_{\lambda_{max}}(P_{max}))$ est elle située en dessous de celle joignant les points $(0, E_0(P_0))$ et $(\lambda_{max}, E_{\lambda_{max}}(P_{max}))$. Notons que les points $(0, E_0(P_0))$ et $(0, 0)$ sont confondus si P_0 est la partition triviale (une région par pixel de l'image d'origine).

Nous obtenons donc pour toute hiérarchie H et tout paramètre d'échelle $\lambda \in [0, \lambda_{max}]$:

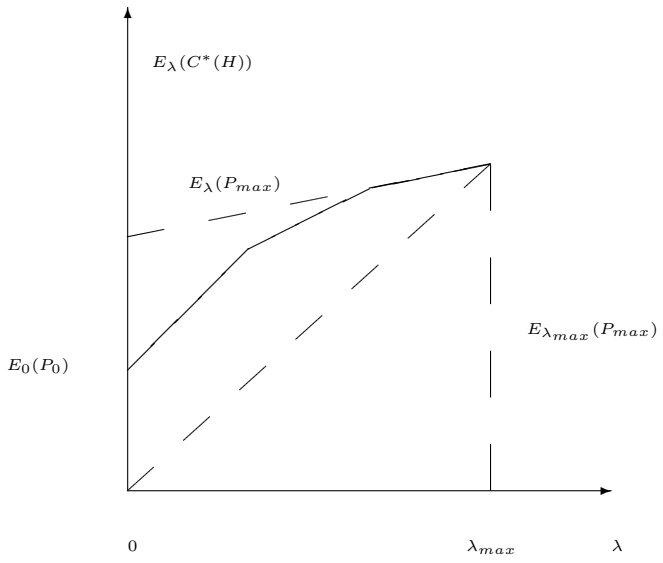


FIG. 3.1 – Encadrement de l'énergie des coupes optimales

$$\frac{\lambda}{\lambda_{max}} E_{\lambda_{max}}(P_{max}) \leq E_\lambda(C^*_\lambda(H)) \leq E_\lambda(P_{max})$$

ou encore :

$$\begin{array}{rcccl}
0 & \leq & E_\lambda(P_{max}) - E_\lambda(C^*_\lambda(H)) & \leq & E_\lambda(P_{max}) - \frac{\lambda}{\lambda_{max}} E_{\lambda_{max}}(P_{max}) \\
0 & \leq & 1 - \frac{E_\lambda(C^*_\lambda(H))}{E_\lambda(P_{max})} & \leq & \dfrac{D_I + \lambda C_I - \frac{\lambda}{\lambda_{max}}(D_I + \lambda_{max} C_I)}{D_I + \lambda C_I} \\
0 & \leq & 1 - \frac{E_\lambda(C^*_\lambda(H))}{E_\lambda(P_{max})} & \leq & D_I \dfrac{1 - \frac{\lambda}{\lambda_{max}}}{D_I + \lambda C_I} \\
0 & \leq & 1 - \frac{E_\lambda(C^*_\lambda(H))}{E_\lambda(P_{max})} & \leq & \dfrac{1 - x_\lambda}{1 + x_\lambda E_I} \quad \text{avec} \left\{ \begin{array}{rcl} x_\lambda & = & \frac{\lambda}{\lambda_{max}} \\ E_I & = & \frac{C_I \lambda_{max}}{D_I} \end{array} \right.
\end{array}$$

La seconde inégalité est basée sur l'hypothèse que $E_\lambda(P_{max}) = D_I + \lambda C_I \geq 0$. On a donc :

$$\begin{array}{rcccl}
1 - \dfrac{1 - x_\lambda}{1 + x_\lambda E_I} & \leq & \frac{E_\lambda(C^*_\lambda(H))}{E_\lambda(P_{max})} & \leq & 1 \\
x_\lambda \dfrac{1 + E_I}{1 + x_\lambda E_I} & \leq & \frac{E_\lambda(C^*_\lambda(H))}{E_\lambda(P_{max})} & \leq & 1 \quad \text{avec } x_\lambda \in [0,1].
\end{array}$$

Nous obtenons finalement :

$$\forall \lambda \in [0, \lambda_{max}] \quad x_\lambda \leq \frac{E_\lambda(C^*_\lambda(H))}{E_\lambda(P_{max})} \leq 1 \tag{3.3}$$

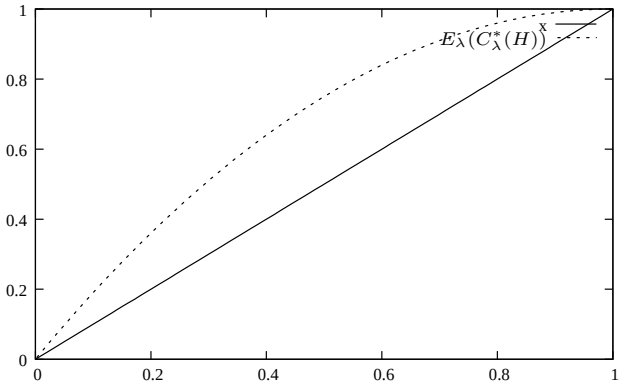


FIG. 3.2 – Représentation d'une énergie normalisée

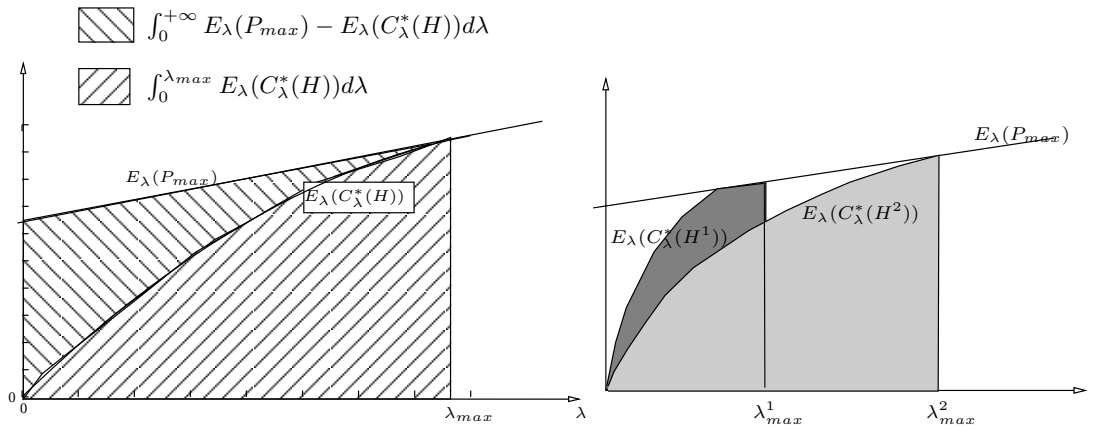
Le terme $E_\lambda(P_{max})$ peut s'interpréter comme l'énergie intrinsèque à l'image et permet d'une certaine façon de mesurer l'adéquation entre notre modèle énergétique et l'image d'entrée. Le rapport $\frac{E_\lambda(C^*_\lambda(H))}{E_\lambda(P_{max})}$ représente donc une énergie normalisée dans laquelle on mesure la qualité (l'énergie) de chaque coupe optimale relativement à l'énergie de l'image globale. L'équation 3.3 montre que cette énergie normalisée se situe au dessus de la diagonale du carré $[0,1]^2$ (Fig. 3.2). La diagonale $y = x$ de $[0,1]^2$ représente le cas limite de courbe concave dans le cube $[0,1]^2$ arrivant sur le point $(1,1)$ et partant de $(0,0)$ ou de $\left(0, \frac{E_0(C^*_0(H))}{D_I}\right)$ si l'attache aux données de la partition initiale n'est pas nulle.

Notons que ce résultat est valide pour n'importe quelle hiérarchie H, obtenue par l'intermédiaire de n'importe quelle heuristique, à partir du moment où le terme C de l'énergie considérée est sous-additif. Cette normalisation de l'énergie nous sera utile lors de l'évaluation des résulats des différentes heuristiques puisqu'elle nous permettra à la fois de comparer différentes heuristiques sur une même image mais également de nous affranchir au moins partiellement de l'influence de l'image initiale pour obtenir des mesures moyennes sur une base d'images.

3.2 Une Heuristique globale

Étant donnée une énergie affine séparable $E(P) = D(P) + \lambda C(P)$ définie pour toute partition P, un algorithme de segmentation hiérarchique produit une hiérarchie H dont la séquence de coupes optimales définit une courbe $E_\lambda(C^*_\lambda(H))$. Contrairement aux approches classiques de minimisation d'énergie où le paramètre λ est fixé, la qualité d'un algorithme de segmentation hiérarchique relativement à une énergie E, doit être basée sur la courbe $E_\lambda(C^*_\lambda(H))$. Cette évaluation doit donc prendre en compte *toutes les échelles* λ.

Un algorithme de segmentation hiérarchique doit donc s'efforcer de minimiser $E_\lambda(C^*_\lambda(H))$ pour tout λ. Une première mesure globale de la qualité d'une segmentation hiérarchique peut être l'aire sous la courbe $E_\lambda(C^*_\lambda(H))$. Toutefois comme la courbe $E_\lambda(C^*_\lambda(H))$ ne converge pas vers 0 à l'infini une telle aire serait infinie. Nous devons donc nous restreindre à l'intervalle $[0, \lambda_{max}]$ où λ_{max} représente l'échelle au delà de laquelle la coupe optimale est la partition triviale (une seule région, Fig. 3.3(a)). Cette mesure représente toutefois un inconvénient puisque un algorithme de segmentation obtenant un λ_{max} petit sera favorisé (Fig. 3.3(b)). Or, au delà de λ_{max} la partition optimale est la partition tri-



(a) critères (b) λ

FIG. 3.3 – Les deux aires que nous pouvons minimiser pour définir une «bonne» segmentation (a). Les énergies de deux hiérarchies H^1 et H^2 avec $\lambda^1_{max} \leq \lambda^2_{max}$ (b).

viale. La valeur λ_{max} représente donc l'intervalle de valeurs d'échelles pour lesquelles l'algorithme de segmentation parvient à produire des partitions non triviales.

Un algorithme de segmentation doit donc simultanément minimiser l'aire sous la courbe $E_\lambda(C^*_\lambda(H))$ pour λ appartenant à $[0, \lambda_{max}]$ et maximiser λ_{max}. Comme l'a suggéré Guigues [GUIGUE03] l'aire comprise entre la droite $E_\lambda(P_{max})$ et $E_\lambda(C^*_\lambda(H))$ (Fig. 3.3(a)) permet de tenir compte de ces deux objectifs simultanément. Notons tout d'abord que ce critère se défini naturellement sur $\mathbb{R}+$ puisque $E_\lambda(C^*_\lambda(H))$ et $E_\lambda(P_{max})$ coïncident au delà de λ_{max}. On a donc :

$$\int_0^{+\infty} E_\lambda(P_{max}) - E_\lambda(C^*_\lambda(H))d\lambda = \int_0^{\lambda_{max}} E_\lambda(P_{max}) - E_\lambda(C^*_\lambda(H))d\lambda$$

De plus, on montre facilement que :

$$\begin{aligned} \int_0^{\lambda_{max}} E_\lambda(P_{max}) - E_\lambda(C^*_\lambda(H))d\lambda &= \int_0^{\lambda_{max}} E_\lambda(P_{max})d\lambda - \int_0^{\lambda_{max}} E_\lambda(C^*_\lambda(H))d\lambda \\ &= (D_I + \tfrac{1}{2}C_I\lambda_{max})\lambda_{max} - \textstyle\sum_{i=1}^{n}(D_i + \tfrac{1}{2}C_i(\lambda_i + \lambda_{i+1}))\Delta\lambda_i \end{aligned} \quad (3.4)$$

où $(D_i)_{i\in\{1,\ ...,n\}}$ et $(C_i)_{i\in\{1,\ ...,n\}}$ représentent respectivement les attaches aux données et les termes de régularisation des n coupes optimales.

La maximisation de l'équation 3.4 passe donc par la maximisation de $(D_I+\frac{1}{2}C_I\lambda_{max})\lambda_{max}$ et la minimisation simultanée de $\sum_{i=1}^{n}(D_i+\frac{1}{2}C_i(\lambda_i+\lambda_{i+1}))\Delta\lambda_i$. Le premier terme est une fonction croissante de λ_{max}. La maximisation de notre critère impose donc de maximiser λ_{max}. Le second terme correspond à l'aire sous la courbe $E_\lambda(C^*_\lambda(H))$. Maximiser notre critère impose donc de simultanément :

- Maximiser λ_{max},
- Minimiser l'aire sous la courbe $E_\lambda(C^*_\lambda(H))$.

Ce qui correspond parfaitement aux contraintes que l'on voulait exprimer.

Des solutions indépendantes à ces deux problèmes d'optimisation sont décrites respectivement dans les sections 3.2.1 et 3.2.2.

3.2.1 Estimation de λ_{max}

Le problème de l'estimation de λ_{max} peut se formuler ainsi : Étant données une partition initiale P_0 et une hiérarchie H construite sur P_0, l'énergie des coupes optimales $E_\lambda(C^*_\lambda(H))$ coupe $E_\lambda(P_{max})$ en un point $(\lambda^H_{max}, E_{\lambda^H_{max}}(P_{max}))$. Il nous faut donc envisager toute les hiérarchies pouvant être construites sur P_0 de façon à sélectionner celle fournissant le λ^H_{max} maximum :

$$\lambda_{max} = \max_{H \in \mathbb{H}(P_0)} \lambda^H_{max}$$

où $\mathbb{H}(P_0)$ représente l'ensemble des hiérarchies construites sur P_0.

L'énumération de toutes les hiérarchies de $\mathbb{H}(P_0)$ étant évidement hors de question nous proposons d'approximer la solution optimale à l'aide d'une heuristique.

Notre première restriction consiste à considérer uniquement des fusions binaires. Dans ce cadre, le nombre de régions de P_0 va diminuer de 1 à chaque fusion jusqu'a ce que l'on obtienne la partition triviale P_{max}. La coupe optimale définie juste avant la partition triviale sera donc composée de deux régions et l'intersection de la droite correspondant à cette coupe avec $E_\lambda(P_{max})$ va déterminer λ_{max}. Le problème de la détermination de λ_{max} revient donc (modulo la restriction aux fusions binaires) à déterminer le regroupement des régions de P_0 en deux «méta-régions» dont la droite associée coupe $E_\lambda(P_{max})$ avec une abcisse λ_{max} maximale.

Notre stratégie pour estimer λ_{max} consiste, dans ce cadre, à fusionner itérativement les couples de régions en prenant explicitement en compte la maximisation de λ_{max}. Pour cela nous calculons pour chaque couple de régions de la partition courante l'attache aux données D et la régularisation C de la partition issue de leur éventuelle fusion. Nous calculons ensuite, l'abcisse λ de l'intersection entre la droite $D + \lambda C$ et $E_\lambda(P_{max})$:

$$\lambda = \frac{D_I - D}{C - C_I} \tag{3.5}$$

Le paramètre λ correspond à la valeur de λ_{max} qui serait obtenue si après avoir fusionné les deux régions envisagées on fusionnait directement toutes les régions restantes pour obtenir la partition triviale P_{max}.

Notre stratégie de fusion consiste donc à maximiser λ_{max} à chaque étape. Cette fusion itérative de couples de régions nous conduit à une partition composée de deux régions. L'intersection de la droite correspondant à cette partition avec $E_\lambda(P_{max})$ définit λ_{max}.

Cette stratégie pose toutefois deux problèmes. Tout d'abord le calcul de λ_{max} n'est pas une fin en soi, mais simplement une étape dans le calcul de la segmentation hiérarchique. Or l'algorithme précédent est a priori aussi long qu'une fusion binaire telle que décrite par Guigues [GUIGUE06]. On peut donc doubler les temps de calcul. De plus, la maximisation de l'intersection entre l'énergie de la partition courante et $E_\lambda(P_{max})$ est un critère global qui a peu de sens lors des premières étapes de fusion.

Afin de répondre à ces deux limitations nous proposons une optimisation du processus précédent lorsque l'énergie d'attache aux données correspond à l'erreur quadratique. Dans ce cas la droite correspondant à la partition en deux régions R_1 et R_2 sera définie par :

$$EQ(R_1) + EQ(R_2) + \lambda(C_1 + C_2)$$

où $EQ(R) = \sum_{x \in R} \|x - \mu_R\|^2$ représente l'erreur quadratique de R.

Cette droite coupera $E_\lambda(P_{max})$ d'autant plus tard que l'ordonnée à l'origine ($EQ(R_1)+EQ(R_2)$) de l'avant dernière partition optimale est peu élevée. On utilise donc un algorithme de quantification [BR00B] qui va regrouper l'ensemble des couleurs de l'image en deux classes C_1 et C_2 telles que $EQ(C_1) + EQ(C_2)$ est minimal. Notre optimisation va consister à fusionner les ensembles connexes de régions dont la moyenne appartient à une même classe. On obtient donc une partition avec un nombre réduit de régions qui approxime, à partir de P_0, la partition de l'image qui serait obtenue en calculant les composantes connexes des deux classes C_1 et C_2(étape d'inversion de table de couleur). Les fusions itératives maximisant la valeur de λ sont effectuées à partir de cette partition.

En résumé notre estimation de λ_{max} suit les étapes suivantes :

1. Calcul de la partition initiale P_0,
2. Quantification de l'ensemble des couleurs de l'image en deux classes,
3. Fusion de tous les ensembles connexes de régions dont la moyenne appartient à une même classe,
4. Fusion itérative des couples de régions restant en fusionnant à chaque étape le couple de régions adjacentes qui maximise l'équation 3.5. Les fusions sont stopées lorsqu'il ne reste plus que deux régions. Appelons P_{n-2} la partition obtenue (avec n le nombre de régions de P_0).
5. Calcul de l'estimation de λ_{max} par intersection de $E_\lambda(P_{n-2})$ et $E_\lambda(P_{max})$.

3.2.2 Interpolation de l'énergie

Comme mentionné dans la section 3.2, la maximisation de l'aire située entre $E_\lambda(P_{max})$ et $E_\lambda(C^*_\lambda(H))$ suppose de maximiser λ_{max} et minimiser l'aire sous la courbe $E_\lambda(C^*_\lambda(H))$. La méthode proposée dans la section précédente permet d'estimer λ_{max}. Toutefois cette méthode nous fournit outre λ_{max}, la partition P_{n-2}. On connait donc :

1. une estimation de λ_{max},
2. la partition optimale en deux régions P_{n-2} à définir pour obtenir λ_{max},
3. la pente C_{n-2} et l'ordonnée à l'origine D_{n-2} de l'énergie $E_\lambda(P_{n-2})$.

Nous comptons utiliser l'ensemble de ces informations pour minimiser l'aire sous $E_\lambda(C^*_\lambda(H))$. Si nous désignons par R_1 et R_2 les deux régions de P_{n-2}, R_1 et R_2 sont construits à partir de P_0. On peut donc définir deux ensembles de régions S_1 et S_2, tels que :

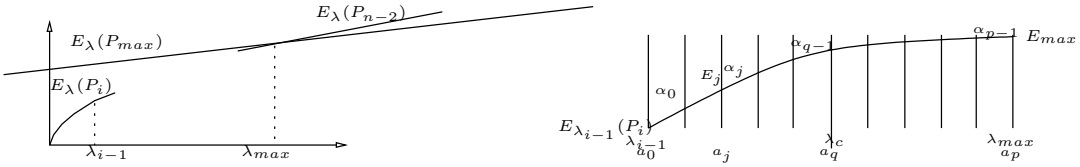
$$R_1 = \bigcup_{R \in S_1} R \text{ et } R_2 = \bigcup_{R \in S_2} R$$

R_1 et R_2 définissant une partition de l'image, S_1 et S_2 définissent une partition de l'ensemble des régions de P_0.

Notre heuristique de construction de la pyramide utilise des fusions binaires restreintes à S_1 et S_2. En d'autres termes, nous nous interdisons de fusionner une région de S_1 avec une région de S_2. Cette restriction permet de garantir que la partition optimale obtenue lorsqu'il ne restera plus que deux régions est égale à P_{n-2} et donc que le λ_{max} associé à cette partition sera égal à celui calculé dans la section 3.2.1.

Après chaque fusion de deux régions dans S_1 et S_2, nous obtenons une partition plus grossière, en appliquant le principe de l'escalade (équation 2.50), nous pouvons déterminer l'échelle d'apparition de cette nouvelle partition. On obtient donc une suite croissante de partitions $P_0, \ldots, P_{n-1} = P_{max}$, chaque partition étant associée à une échelle d'apparition $(\lambda_i)_{i\in\{0,\ldots,n-1\}}$. Le terme de régularisation C étant sous additif, la suite $(\lambda_i)_{i\in\{0,\ldots,n-1\}}$ est croissante si $P_0, \ldots, P_{n-1}$ approxime efficacement les coupes optimales. Nous supposerons dans la suite que cette condition est effectivement réalisée.

Le choix des deux régions à fusionner dans S_1 et S_2 est réalisé grâce à un algorithme itératif qui utilise les informations de la partition P_{n-2} pour déterminer les deux régions à fusionner. Supposons que nous ayons calculé la suite $P_0, \ldots, P_{n-1}$ jusqu'à la partition P_i, $i \in \{0, \ldots, n-3\}$. Si nous approximons $E_\lambda(C^*_\lambda(H))$ par la suite $P_0, \ldots, P_{n-1}$, la courbe $E_\lambda(C^*_\lambda(H))$, passera par le point $(\lambda_{i-1}, E_{\lambda_{i-1}}(P_i))$ avec une pente égale à C_i à droite de λ_{i-1}. La courbe $E_\lambda(C^*_\lambda(H))$ arrivera également en λ_{max} avec une pente égale à C_{n-2} à gauche de λ_{max}(Fig. 3.4(a)). Nous connaissons donc deux points et deux pentes de la courbe $E_\lambda(C^*_\lambda(H))$ et savons de plus que cette courbe est concave et linéaire par morceaux. On peut donc estimer l'aire de $E_\lambda(C^*_\lambda(H))$ entre λ_{i-1} et λ_{max} à l'aide d'une interpolation de la courbe entre ces deux points.



(a) Construction de la hiérarchie (b) Interpolation

FIG. 3.4 – Représentation des données connues lors de la construction de la hiérarchie (a) et illustrations des principales notations lors de l'interpolation(b).

Le problème de l'interpolation étant largement sous contraint nous utilisons la modélisation suivante(Fig. 3.4(b)) :

1. L'intervalle $[\lambda_{i-1}, \lambda_{max}]$ est décomposé en deux sous intervalles $[\lambda_{i-1}, \lambda_c]$ et $[\lambda_c, \lambda_{max}]$. Nous avons arbitrairement fixé λ_c à $\frac{\lambda_{i-1}+\lambda_{max}}{2}$.

2. Si $p = n - i$ désigne le nombre de régions de la partition courante on divise chaque intervalle $[\lambda_{i-1}, \lambda_c]$ et $[\lambda_c, \lambda_{max}]$ en $q = [\frac{p}{2}]$ sous intervalles $I_j = [a_j, a_{j+1}]$ avec $a_0 = \lambda_0$, $a_q = \lambda_c$ et $a_p = \lambda_{max}$.

3. On modélise $E_\lambda(C^*_\lambda(H))$ sur chaque intervalle I_j par une droite $E_j + \alpha_j(\lambda - a_j)$. On a $\alpha_0 = C_i$ et $\alpha_{p-1} = C_{n-2}$. On suppose de plus que les pentes α_j décroissent par pas de ϵ_1 et ϵ_2 respectivement sur $[\lambda_{i-1}, \lambda_c]$ et $[\lambda_c, \lambda_{max}]$.

$$\begin{cases} \alpha_i &= \alpha_0 - i\epsilon_1, & \forall i \in \{0, \ldots, q-1\} \\ \alpha_i &= \alpha_{q-1} - (i-q+1)\epsilon_2, & \forall i \in \{q, \ldots, p-1\} \end{cases}$$

Ces trois contraintes permettent de déterminer ϵ_1 et ϵ_2 et déterminent donc l'interpolation de $E_\lambda(C^*_\lambda(H)$ sur $[\lambda_{i-1}, \lambda_{max}]$. On montre (Annexe, section A.1) que l'aire sous la courbe interpolée est égale à :

$$\begin{aligned}\mathcal{A} = \sum_{j=0}^{p-1} \int_{a_j}^{a_{j+1}} E_i + \alpha_j(\lambda - a_j)d\lambda \quad &= \quad q\Delta_1 E_{\lambda_{i-1}}(P_i) + \tfrac{1}{2}\alpha_0\Delta_1^2 q^2 - \Delta_1^2\epsilon_1\tfrac{q(q-1)(2q-1)}{12} \\ &\quad +(p-q)\Delta_2 E_q + \tfrac{1}{2}\Delta_2^2(p-q)^2\alpha_{q-1} \\ &\quad -\tfrac{1}{12}(p-q)\epsilon_2\Delta_2^2\left[2(p-q)^2 + 3(p-q) + 1\right]\end{aligned}$$

où E_q est la valeur de la courbe interpolée en λ_c. Les symboles Δ_1 et Δ_2 représentent respectivement la largeur des intervalles I_j sur $[\lambda_{i-1}, \lambda_c]$ et $[\lambda_c, \lambda_{max}]$.

$$\Delta_1 = \frac{\lambda_c - \lambda_{i-1}}{q} \text{ et } \Delta_2 = \frac{\lambda_{max} - \lambda_c}{q}, E_q = E_0 + q\Delta_1\left(\alpha_0 + \frac{1}{2}\epsilon_1\right) - \frac{q^2}{2}\Delta_1\epsilon_1$$

Nous obtenons ainsi une expression formelle de l'intégrale en fonction de $n, i, \lambda_{i-1}, \lambda_{max}$, $E_{\lambda_{i-1}}(P_i), E_{\lambda_{max}}(P_{max})$ qui sont supposés connus. Nous pouvons ainsi définir un algoritme permettant de sélectionner les régions à fusionner, en se basant uniquement sur un critère global. A chaque niveau, ce processus de décimation nécessite d'effectuer les étapes suivantes :

1. Fixer un λ_c et en déduire les valeurs de $\Delta_1, \epsilon_1, \Delta_2, \epsilon_2$.
2. Calculer la valeur de l'intégrale $\mathcal{A}$ pour chacune des fusions envisageable et fusionner les régions impliquant la plus petite intégrale.
3. Itérer ce processus jusqu'à ce que toutes les régions soient fusionnées

Ce critère de fusion basé sur un critère de minimisation global n'a cependant pas donné de résultats concluants d'un point de vue qualitatif. Nous pensons que ces limitations viennent d'une part du grand nombre d'hypothèses nécessaires à l'interpolation de $E_\lambda(C^*_\lambda(H))$ et d'autre part du fait que la minimisation de l'aire est un critère très global qui n'est sans doute pas pertinent lorsque l'on effectue les premières fusions et que l'on est loin de λ_{max}. Ceci nous a amené à nous pencher plus en détails sur des heuristiques de fusion utilisant des critères de fusion locaux et permettant d'élargir l'espace de recherche.

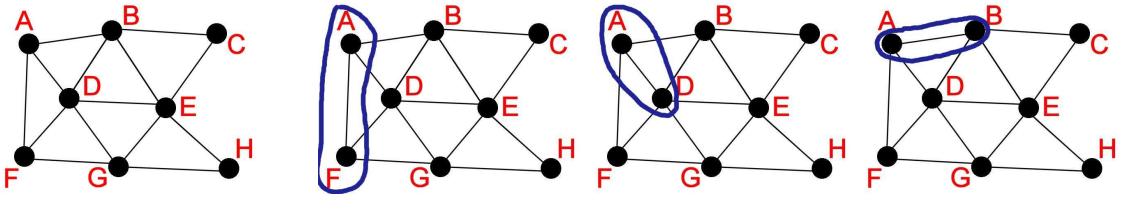
3.3 Heuristiques locales

3.3.1 Fusion séquentielle

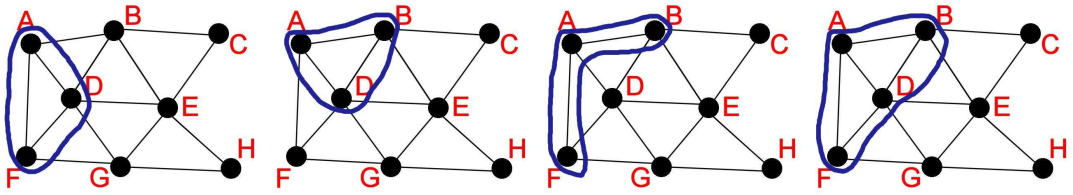
Pour une partition P donnée, considérons pour chaque région $R \in P$ l'ensemble $V(R)$ des régions qui lui sont adjacentes. L'ensemble $V(R)$ des régions adjacentes à une région R est défini de la façon suivante grâce à la relation d'adjacence $\sim$:

$$\forall R \in P \quad V(R) = \{X | X \sim R\} \tag{3.6}$$

Par exemple sur la Figure 3.5(a), représentant le graphe d'adjacence de régions d'une partition P, l'ensemble des voisins de la région E est noté $V(E) = \{E, B, C, D, G, H\}$. Notons qu'avec cette définition, une région appartient toujours à son voisinage.



(a) RAG G (b) $\{A, F\}$ (c) $\{A, D\}$ (d) $\{A, B\}$



(e) $\{A, F, D\}$ (f) $\{A, B, D\}$ (g) $\{A, B, F\}$ (h) $\{A, B, D, F\}$

FIG. 3.5 – Représentation de l'ensemble $\mathcal{P}(V(A))$ (b) à (h) sur le RAG G (a)

Le but de cette approche étant de réduire l'énergie des coupes optimales dans la hiérarchie finale, le processus de décimation va élargir l'espace de recherche par rapport à la construction d'une hiérarchie binaire.

A chaque itération une région peut fusionner avec plusieurs régions incluses dans son voisinage.

Nous définissons alors l'ensemble des sous-ensembles de $V(R)$, noté $\mathcal{P}(V(R))$ de la façon suivante :

$$\forall R \in P \quad \mathcal{P}(V(R)) = \{\{R\} \cup X | X \subseteq V(R) \setminus \{R\}\} \tag{3.7}$$

Notons que le cardinal de l'ensemble $\mathcal{P}(V(R))$ est égal au cardinal de l'ensemble des parties de $V(R) - \{R\}$ moins l'ensemble vide. En effet le sommet central appartient toujours à l'ensemble de sommets à fusionner. De plus, le sous ensemble trivial réduit à $\{R\}$ ne code aucune fusion est n"est donc pas pris en compte. Le cardinal de l'ensemble des parties d'un ensemble de n éléments étant égal à 2^n, si $Card(V(R)) = N$ alors $Card(\mathcal{P}(V(R))) = 2^{N-1} - 1$.

La Figure 3.5 illustre l'équation 3.7 en présentant l'ensemble des sous-ensembles du voisinage $V(A)$ du sommet A dans le graphe G (Figure 3.5(a)). Dans cet exemple $V(A) = \{A, B, D, F\}$ et $\mathcal{P}(V(A)) = \{\{A, B\}, \{A, D\}, \{A, F\}, \{A, B, D\}, \{A, B, F\}, \{A, B, D, F\}\}$. Nous pouvons remarquer que $|V(R)| = 4$ et que $|\mathcal{P}(V(R))| = 2^3 - 1 = 7$.

Notons $W \in \mathcal{P}(V(R))$ l'un des sous-ensembles de $\mathcal{P}(V(R))$ et $R^W = \bigcup_{R' \in W} R'$ la région formée de l'union des régions de l'ensemble W. $W \in \mathcal{P}(V(R))$ représentent ainsi une des fusions possibles de la région R avec au moins l'un de ses voisins.

Si nous considérons deux partitionnements de R^W induits par W : $\begin{cases} P_{R^W} = \{R^W\} \\ P_W = W \end{cases}$

$P_{R^W} = \{R^W\}$ représentant la sur-partition absolue de l'ensemble R^W et $P_W = W$ représentant la sous partition absolue de R^W. Les énergies respectives associées à chacunes de ses partitions sont définies.

$$\begin{cases} E_\lambda(P_{R^W}) &= D(R^W) + \lambda C(R^W) \\ E_\lambda(P_W) &= D(W) + \lambda C(W) = \sum_{R' \in W} D(R') + \lambda \sum_{R' \in W} C(R') \end{cases} \quad (3.8)$$

avec $D(W)$ représentant le terme d'attache aux données et $C(W)$ représentant le terme de régularisation pour la partition P_W. Si l'énergie C est une énergie sous-additive (voir Déf.66) nous avons :

$$C(R^W) < C(W)$$

- Si $D(R^W) < D(W)$ alors l'énergie de la partition P_{R^W} formée de l'union de régions de l'ensemble W est toujours inférieure à l'énergie de la partition P_W.
- Sinon, si $D(W) < D(R^W)$, l'énergie $E_\lambda(P_W)$ de la partition P_W est inférieure à l'énergie $E_\lambda(P_{R^W})$ de la partition P_{R^W} pour des valeurs du paramètre d'échelle λ inférieures à :

  $$\lambda^+(R^W) = \frac{D(R^W) - D(W)}{C(W) - C(R^W)},$$

  caractérisant l'échelle d'apparition de la région R^W. En se basant sur le principe de l'escalade défini dans la section 2.6.4, cet algorithme séquentiel va calculer pour chaque région R de la partition P l'échelle d'apparition minimale relative à une région R^W :

  $$\lambda^+_{min}(R) = \arg min_{W \in \mathcal{P}^*(V(R))} \frac{D(R^W) - D(W)}{C(W) - C(R^W)} \quad (3.9)$$

  L'ensemble $W \in \mathcal{P}^*(V(R))$ qui réalise le minimum est noté $W_{min}(R)$. C'est cet ensemble qui devra fusionner de préférence pour minimiser l'énergie de la partition résultante.

Nous utilisons donc l'échelle d'apparition λ^+ pour définir notre algorithme séquentiel. Pour une énergie E donnée, cet algorithme réalise les étapes suivantes de façon itérative :

1. Soit P la partition courante initialisée avec une partition initiale P_0,
2. Pour toutes les régions R appartenant à la partition P, calculer son échelle d'apparition $\lambda^+_{min}(R)$ ainsi que le minimum $W_{min}(R)$.
3. calculer $R_{min} = \arg min_{R \in P} \lambda^+_{min}(R)$ et fusionner toutes les régions appartenant à l'ensemble $W_{min}(R_{min})$.
4. Si la partition compte encore plus de deux régions retourner à l'étape 2,
5. Renvoyer la hiérarchie finale encodant la séquence des opération de fusions successivement effectuées.

Cet algorithme effectue donc, à chaque étape, une seule fusion pouvant faire intervenir de 2 à k régions (k étant borné par le nombre maximum de voisins d'une région).

La Figure 3.6 illustre une étape de cette stratégie d'escalade n-aire effectuée sur une pré-hiérarchie composée de trois sommets E,G et H. Toutes les fusions possibles entre le sommet H et ses voisins sont envisagées. Les fusions $G \cup H$, $E \cup H$ et $E \cup G \cup H$ sont alors toutes considérées, pour ensuite sélectionner celle qui possède la plus petite échelle d'apparition λ^+, dans cet exemple on retient $H \cup G$.

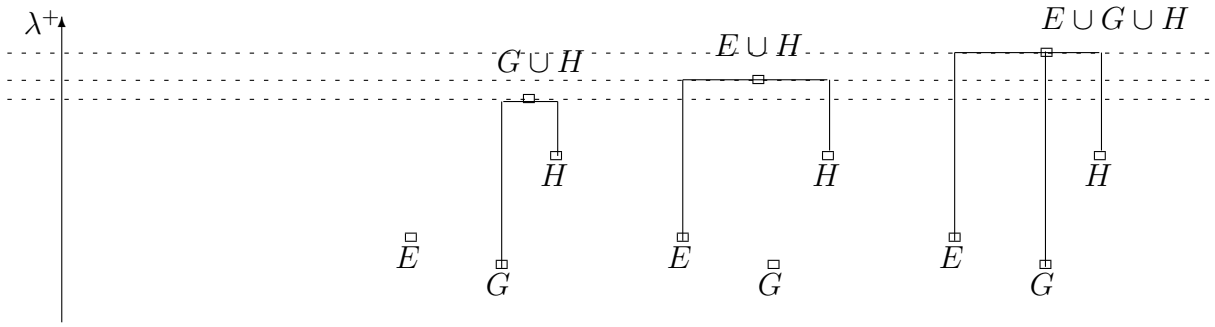


FIG. 3.6 – Exemple de calcul de l'ensemble $W_{min}(H)$. Dans cette exemple $V(H) = \{E, G\}$, l'ensemble $\mathcal{P}(V(H)) = \{G \cup H, E \cup H, E \cup G \cup H\}$ et l'ensemble $W_{min}(H) = G \cup H$. La fusion retenue serait celle impliquant les sommets H et G.

Puisque toutes les régions appartenant à l'ensemble $W_{min}(R_{min})$ sont adjacentes à la région R_{min}, dans le cadre des pyramides irrégulières, cette opération de fusion peut être codée à l'aide d'un noyau de contraction (voir Déf 52) de profondeur un, composé d'un unique arbre dont la racine est égale à la région R_{min}.

Du point de vue de la complexité, le calcul de l'échelle d'apparition $\lambda^+_{min}(R)$ pour chaque région R de la partition nécessite l'exploration de l'ensemble $\mathcal{P}^*(V(R))$ dont le cardinal est égal à $2^{|V(R)|-1}$. Si l'on considère le graphe $G = (V, E)$ codant la partition P, la complexité de chacune des étapes de l'algorithme peut donc être bornée par $\mathcal{O}(|V|2^k)$ avec $|V|$ représentant le nombre de sommets du graphe G (équivalant au nombre de régions de la partition P) alors que k représente l'arité maximum des sommets de G.

Le cardinal de V décroît à chaque itération de $|W_{min}(R_{min})| - 1$. Notons que comme le cardinal de $W_{min}(R_{min})$ est au moins égal à deux, le cardinal de V diminue au moins de un à chaque étape. Cet algorithme se termine donc en un temps fini et sa complexité globale peut être bornée par $\mathcal{O}(|V|^2 2^k)$ avec $|V|$ représentant le nombre de sommets du graphe G et k le degré maximum des sommets de G.

Nous pouvons également noter que la méthode d'escalade binaire proposée par Guigues (voir Section 2.6.4) est incluse dans cet algorithme. En effet l'opération de fusion binaire correspond simplement à un sous-ensemble $W_2 \in \mathcal{P}^*(V(R))$ avec le cadinal $|W_2| = 2$ de l'ensemble de solutions considéré par cet algorithme.

3.3.2 Fusion parallèle

Les algorithmes basés sur une décimation séquentielle, présentés notamment dans la Section 3.3.1, peuvent s'avérer très coûteux en temps, même dans le cas ou l'on borne le nombre de régions à fusionner à chaque étape à deux, comme dans le cas de l'escalade binaire (voir Section 2.6.4).

Non plus dans l'optique d'obtenir une énergie de partition toujours plus faible, mais plutôt dans le but d'accélérer le temps d'éxecution nous nous sommes intéressés à la parallélisation de l'algorithme. Nous avons donc défini un algorithme de fusion de régions parallèle, basé sur la notion de couplage maximal (défini Section 2.6.3).

Rappelons qu'un ensemble d'arêtes M d'un graphe $G = (V, E)$ est appelé un couplage maximal si chaque sommet V de G est incident à au moins une arête de l'ensemble M et si M est maximale par rapport à cette propriété (voir Déf 68).

Le but de cet algorithme est donc , à partir d'un graphe $G = (V, E)$ représentant une partition P, de construire un couplage maximal M de façon à ce que l'échelle d'apparition des régions produites par la contraction des arêtes de l'ensemble M soit la plus petite possible.

Notons $\iota(e)$ l'ensemble composé des deux sommets incidents à l'arête e. Nous allons utiliser une approche similaire à celle utilisée dans la Section 3.3.1 mais en affectant désormais une échelle d'apparition $\lambda^+(e)$ à chacune des arêtes de G. Si $\iota(e) = \{R, R'\}$ alors l'échelle d'apparition λ^+ peut s'écrire :

$$\lambda^+(e) = \lambda^+(R \cup R') = \frac{D(R \cup R') - D(R) - D(R')}{C(R) + C(R') - C(R \cup R')} \tag{3.10}$$

En s'inspirant de la méthode proposée par Haxhimusa [HAXHIM02A] (Section 2.6.3), nous définissons le couplage maximal comme le résultat de la construction d'un noyau (voir Section. 2.6.1) sur l'ensemble des arêtes du graphe G. La construction de cet ensemble est réalisée par le biais d'un processus itératif qui sélectionne à chaque étape les arêtes dont l'échelle d'apparition se trouve être localement minimale. Nous allons utiliser, pour formaliser ce processus itératif, deux variables booléennes p et q attachées à chacune des arêtes, telles que :

$$\left\{ \begin{array}{lcl} p_e^1 & = & \lambda^+(e) = min_{e' \in \Gamma(e)}\{\lambda^+(e')\} \\ q_e^1 & = & \bigwedge_{e' \in \Gamma(e)} \overline{p_{e'}^1} \end{array} \right. \tag{3.11}$$

et

$$\left\{ \begin{array}{lcl} p_e^{k+1} & = & p_e^k \vee \left(q_e^k \wedge \lambda^+(e) = min_{e' \in \Gamma(e) \mid q_{e'}^k}\{\lambda^+(e')\} \right) \\ q_e^{k+1} & = & \bigwedge_{e' \in \Gamma(e)} \overline{p_{e'}^{k+1}} \end{array} \right. \tag{3.12}$$

avec $\Gamma(e)$ représentant le voisinage de l'arête e défini comme :

$$\Gamma(e) = \{e\} \cup \{e' \in E | \iota(e) \cap \iota(e') \neq \emptyset\}$$

.

Si l'on considère par exemple l'arête d'étiquette 17 sur la figure 3.7, son voisinage sera composé de l'ensemble des arêtes incidentes aux deux sommets placés de part et d'autre de cette arête. Ce voisinage sera donc composé des arêtes 5,3,4,1,2 et 7 ainsi que l'arête 17.

Ce processus itératif s'achève lorsqu'aucun changement n'est intervenu entre deux itérations successives. Si n représente la dernière itération, l'ensemble des arêtes tel que p_e^n est $vrai$, définit un couplage maximal M encodant l'ensemble des arêtes à contracter.

Si l'on interprète ce processus en termes de segmentation et que l'on considère $\lambda^+(e)$ comme un score de fusion, une arête e entre deux sommets sera marquée à ($p_e^k = vrai$) à

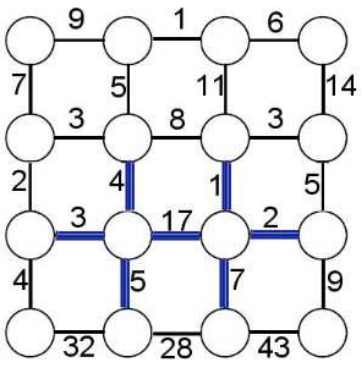


FIG. 3.7 – Ilustration de la notion de voisinage pour une arête. Les arêtes en bleu (trait épais) correspondent aux arêtes incluses dans le voisinage de l'arête 17

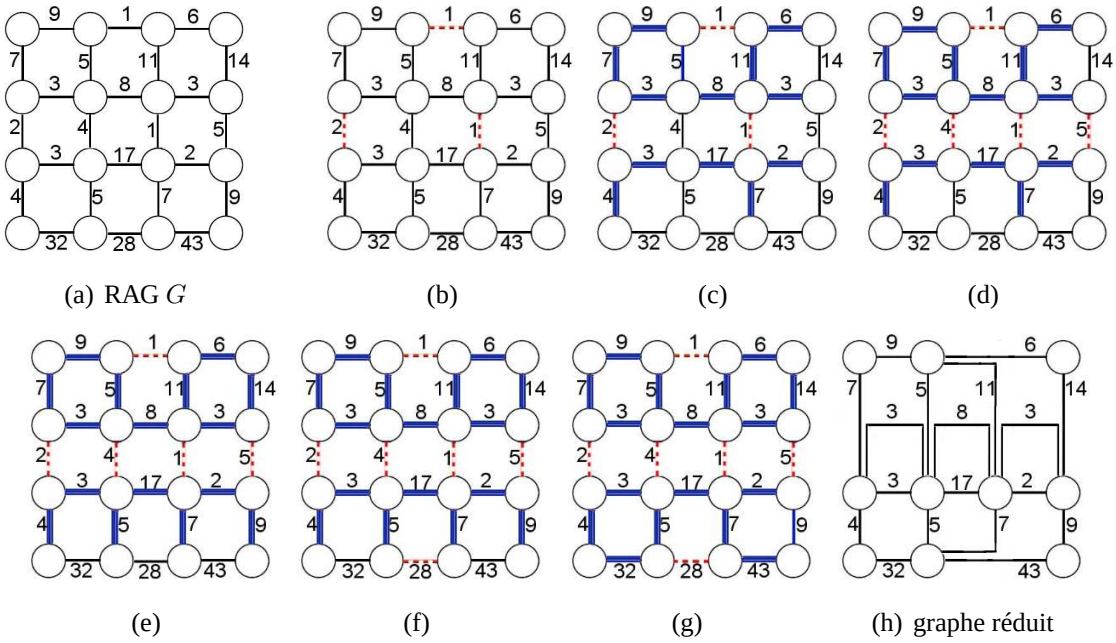
l'itération k, si parmi toutes les possibilités de fusions faisant intervenir ces deux sommets, la fusion faisant intervenir l'arête e obtient le meilleur score. Les deux sommets de part et d'autre de l'arête e vont donc préférer fusionner entre eux, plutôt que de fusionner avec n'importe quel autre sommet présent dans leurs voisinages.

La Figure 3.8 illustre le processus itératif permettant la construction de ce couplage maximal à partir du graphe $G = (V, E)$, présenté sur la Figure 3.8(a). Sur la Figure 3.8(b) seuls les minima locaux sont sélectionnés (en pointillés). La deuxième étape (Figure 3.8(c)) consiste à marquer les arêtes (en gras sur la Figure 3.8) adjacentes aux minima calculés à l'étape précédente, leur interdisant ainsi d'être contractés. Les étapes suivantes itèrent simplement ce processus jusqu'à ce qu'aucun changement n'interviennent entre des étapes successives (dans notre exemple jusqu'à ce que toutes les arêtes soient coloriées en rouge (pointillés) ou en bleu (trait épais)).

Notons que la construction du couplage maximal est seulement la première étape de l'algorithme proposé par Haxhimusa. Pour obtenir un taux de décimation plus stable, au moins égal à deux, à chaque étape de son algorithme, Haxhimusa complète le couplage maximal initialement obtenu en ajoutant des arêtes qui ne sont pas localement minimales.

Plutôt que de favoriser un facteur de réduction constant, nous avons préféré favoriser le critère énergétique. Les arêtes qui sont ainsi contractées, sont celles qui deviennent localement minimales à une itération du processus de décimation. Comme l'a démontré Bield [BIEDL04] le facteur de réduction, en terme d'arêtes induit par l'utilisation d'un couplage maximum est au minimum égal à $2\frac{k-1}{2k-1}$ avec k le degré maximum des sommets du graphe courant. Le taux de décimation des arêtes peut donc être très petit pour des graphes dont l'arité des sommets est très importante. Nous verrons néanmoins dans la Section 3.4 que expérimentalement, le taux de décimation moyen des arêtes entre chaque niveau, testé sur une base de 100 images, est égal à 1.73. Ce taux est comparable au taux de décimation de 2 obtenu par Haxhimusa.

Pour améliorer l'énergie des partitions obtenues par cet algorithme de décimation parallèle, nous avons envisagé une méthode alternative consistant à contracter uniquement les arêtes sélectionnées lors de la première itération (pour $p_e^1 = vrai$). En effet les minimas sélectionnés dans l'équation 3.12 sont calculés sur un ensemble dont la taille diminue au fur et à mesure des itérations pour compléter le couplage maximum. Les minimas détectés sont donc de moins en moins significatifs au fur et à mesure des itérations et certaines



(a) RAG G (b) (c) (d)

(e) (f) (g) (h) graphe réduit

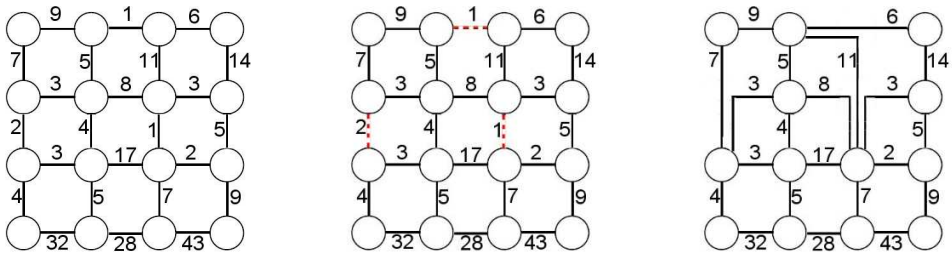
FIG. 3.8 – Processus de construction itératif de couplage maximal sur un graphe G (a). Les arêtes rouges (en pointillés) correspondent aux arêtes qui seront contractées, les bleues (en trait épais) correspondent aux arêtes ne pouvant être contractées du fait qu'elles se trouvent dans le voisinage d'une arête non survivante (b) à (g). Le graphe réduit est présenté sur la figure (g).

fusions qui ne seraient pas effectuées de façon naturelle sont contraintes, ce qui revient en quelque sorte à forcer certaines fusions. Le fait de ne contracter que les arêtes sélectionnées à la première itération, revient à calculer uniquement les minima sur le voisinage de chaque arête $e \in G$. Cette seconde heuristique de fusion peut être interprétée comme une combinaison de la méthode proposée par Haxhimusa [HAXHIM02A] (Section 2.6.3) mixée avec la méthode de décimation stochastique guidée par les données proposée par Jolion [JOLI01] (Section 2.6.2), qui consiste à fusionner immédiatement les sommets correspondant à des minima locaux.

Pour une énergie E donnée, cet algorithme réalise les étapes suivantes de façon itérative :

1. Soit P la partition courante initialisée avec une partition initiale P_0,
2. Pour toutes les arêtes e appartenant à la partition P, calculer son échelle d'apparition $\lambda^+(e)$.
3. sélectionner toutes les arêtes étant localement optimales et fusionner toutes les régions de part et d'autre de ces arêtes.
4. Si la partition compte encore plus de deux régions retourner à l'étape 2,
5. Renvoyer la hiérarchie finale encodant la séquence des opérations de fusions successivement effectuées.

Cet algorithme est illustré sur la figure 3.9. Les arêtes sélectionnées, qui vont immédiatement être contractées, correspondent aux arêtes sélectionnées à la première étape de l'algorithme décrit sur la figure 3.8 Cette approche ne contraint aucune fusion à être ef-



(a) RAG G (b) arêtes minimales à contracter (c) graphe réduit

FIG. 3.9 – Une étape de l'algorithme de décimation MM^1 sur le graphe initial G (a). Les minima locaux représentés en rouge (pointillés) (b) sont contractés pour obtenir le graphe réduit (c).

fectuée. Le taux de décimation moyen obtenu pour cet algorithme est bien sur inférieur au taux de décimation de l'algorithme basé sur le couplage maximal. Le nombre de niveaux de la pyramide sera donc plus élevé, mais le temps de construction nécessaire pour construire chaque niveau est moins important. Nous verrons dans la Section 3.4 que cette approche permet d'accélérer la construction de la hiérarchie tout en maintenant des énergies de partition relativement proches de celles obtenues en utilisant la méthode basée sur le couplage maximum.

3.4 Évaluation

Nous allons présenter dans cette section des résultats et des évaluations aussi bien qualitatives que quantitatives sur l'ensemble des algorithmes présentés dans les section 3.3.1 et 3.3.2. Les différentes heuristiques présentées on été évaluées sur les 100 images composant la base de données d'images naturelles de Berkeley (Figure 3.10).

Pour simplifier les notations nous allons utiliser les acronymes suivants pour les différentes heuristiques proposées. Nous appellerons (MM) (pour *Maximal Matching*) l'heuristique de fusion basée sur le couplage maximal et (MM^1) la variation de cette méthode consistant à fusionner à chaque étape les arêtes sélectionnées lors de la première itération (voir Section 3.3.2). L'heuristique séquentielle présentée à la section 3.3.1, permettant à une région de fusionner avec tout ou une partie de son voisinage, sera appelée (SM) (pour *Sequential Merging*). Nous allons également étudier deux variantes de l'algorithme (SM), l'une appelée (SM^5), consistant à borner le nombre maximum de voisins à fusionner à 5, et l'autre appelée (SM^2) restreignant le cardinal de l'ensemble de régions à fusionner à deux. Cette dernière heuristique correspond à l'escalade binaire proposée par Guigues [GUIGUE06] (voir Section 2.6.4).

Ces algorithmes peuvent s'adapter à tout type de partition initiale. Dans un souci de cohérence, l'ensemble des expérimentations présentées dans cette section utilisent une sursegmentation initiale obtenue par l'intermédiaire d'un algorithme de *ligne de partage des eaux* défini sur les cartes combinatoires par Luc Brun [BRUN05B].

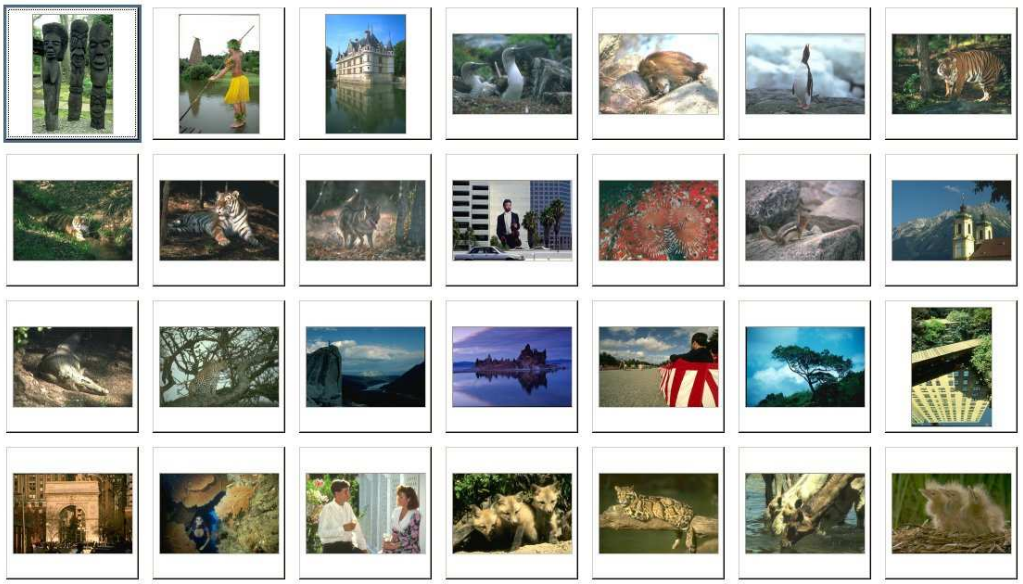


FIG. 3.10 – Illustration présentant une partie de la base de Berkeley proposant 100 images naturelles (cette base est disponible à l'adresse http ://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/).

De nombreuses énergies ont été proposées (Section 2.5) permettant de coder différents types de critères d'homogénéité (modèles constant par morceaux [LECLER89, MUMFOR89],variations linéaires ou polynomiales [LECLER89]). Nous allons dans cette section nous restreindre au modèle constant par morceaux défini par Leclerc [LECLER89] et Mumford & Shah [MUMFOR89].

L'énergie relative à ce type de modèle, que nous allons utiliser pour l'ensemble des expériences peut s'écrire

$$E_\lambda(P) = D(P) + \lambda C(P) = \sum_{i=1}^{n} SE(R_i) + \lambda|\delta(R_i)| \tag{3.13}$$

avec

- $P = \{R_1, \ldots, R_n\}$ représentant la partition de l'image
- $SE(R_i) = \sum_{p \in R} \|c_p - \mu_R\|^2$ codant l'erreur quadratique relative à la région R_i
- $|\delta(R_i)|$ représentant la longueur totale des frontières de cette région.

Du fait de la grande variabilité de la forme de la fonction représentative de l'énergie $E_\lambda(C^*_\lambda(H))$, pouvant dépendre à la fois des performances intrinsèques de l'algorithme permettant de construire la hiérarchie H mais également de l'image à partir de laquelle la hiérarchie H va être construite, nous avons procédé à la normalisation des énergies $E_\lambda(C^*_\lambda(H))$ produites par les différentes heuristiques, comme étape préalable, avant d'effectuer des comparaisons.

Nous avons utilisé la normalisation définie dans la section 3.1. Le fait d'utiliser l'énergie normalisée $\frac{E_\lambda(C^*_\lambda(H))}{E_\lambda(P_{max})}$ ainsi qu'un paramètre d'échelle normalisé $x_\lambda = \frac{\lambda}{\lambda_{max}}$, nous assure que toute courbe $\frac{E_\lambda(C^*_\lambda(H))}{E_\lambda(P_{max})}$ sera située dans la partie supérieure gauche du carré unitaire $[0,1]^2$ (voir Fig. 3.1) et nous permet ainsi de borner l'énergie mais également le

paramètre d'échelle sur l'intervalle $[0, 1]$. L'utilisation de cette énergie normalisée permet de réduire l'influence de la variation globale des images sur l'énergie mais également de comparer des énergies obtenues avec la même heuristique sur différentes images.

3.4.0.a Heuristiques de fusions séquentielles

Nous avons dans un premier temps comparé les résultats obtenus en utilisant l'algorithme séquentiel (SM) ainsi que ses variantes (SM^2) et (SM^5) (voir Section 3.3.1).

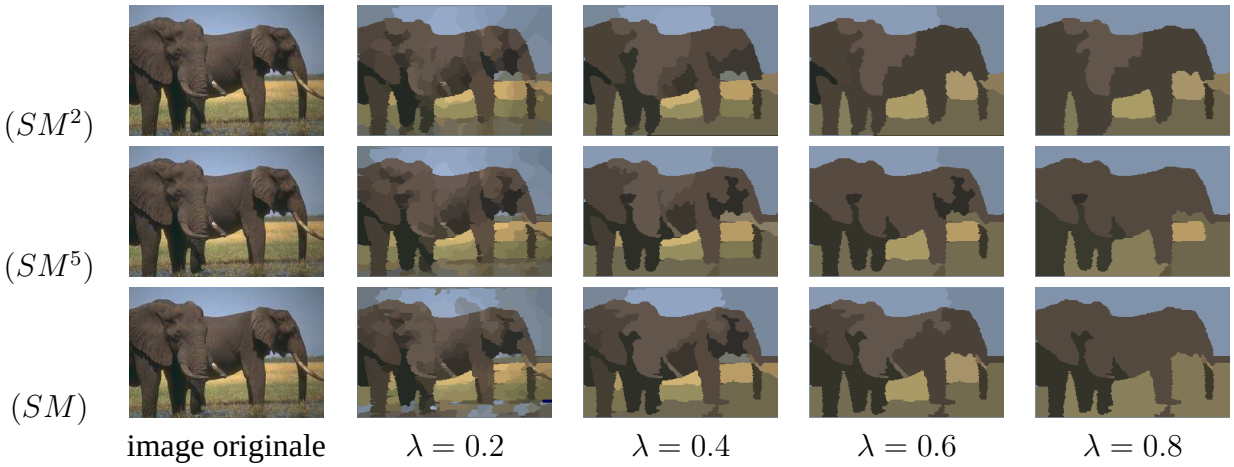


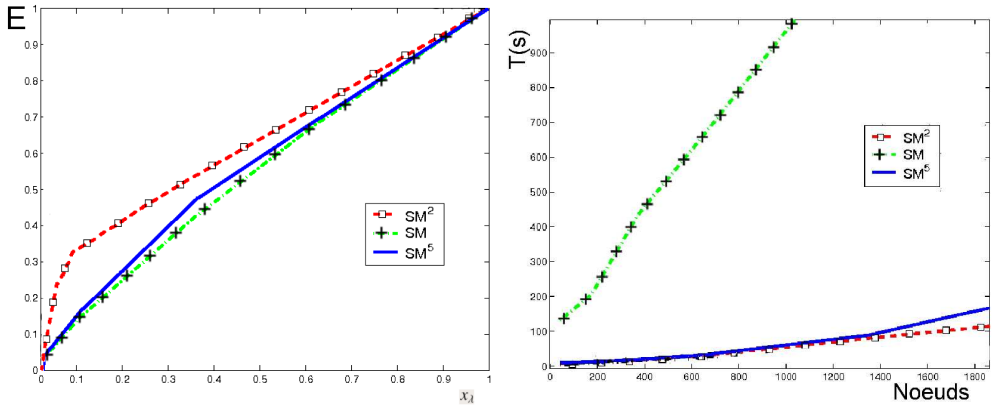
FIG. 3.11 – Partitions de l'image des éléphants à différentes échelles. Chaque ligne de ce tableau correspond à une heuristique séquentielle dont l'acronyme est indiqué dans la première colonne. Les colonnes correspondent aux différentes échelles, le paramètre d'échelle est indiqué en bas pour chacune des colonnes.

La figure 3.11 présente les partitions de l'image des éléphants, obtenues pour chacune des heuristiques, pour différents paramètres d'échelle. Nous pouvons remarquer que visuellement les résultats sont assez similaires pour les échelles les plus fines. L'algorithme (SM) permet de conserver des détails assez fins comme la trompe de l'éléphant dans les échelles les plus grossières.

La figure 3.12 présente les énergies des coupes optimales $E_\lambda(C^*_\lambda(H))$ en fonction du paramètre d'échelle pour chacune des heuristiques de fusion séquentielle ainsi que les temps d'exécution nécessaires à la construction des hiérarchies en fonction du nombre de régions présentes dans la partition initiale. Ces courbes sont des résultats moyents obtenus sur l'ensemble de la base d'images. L'énergie relative à l'algorithme (SM) est toujours bien inférieure à celle produite par l'algorithme (SM^2) (correspondant à l'escalade binaire proposée par Guigues) (Fig. 3.12(a)). Le fait d'élargir l'espace de recherche permet donc de réduire de façon significative l'énergie des partitions produites. En revanche le temps d'exécution de (SM) explose littérallement par rapport à celui de (SM^2) (figure 3.12)

Nous pouvons néanmoins observer que les énergies correspondant à l'algorithme (SM) et à sa restriction (SM^5), consistant à restreindre le cardinal de l'ensemble de régions à fusionner à cinq, sont très proches. En revanche le temps d'exécution de l'algorithme (SM^5) est quelque peu supérieur à celui de (SM^2) mais largement inférieur à celui de (SM).

Le fait de limiter le cardinal du nombre de régions à fusionner à 5 semble apparaître comme un excellent compromis permettant d'obtenir des énergies faibles tout en conservant un temps de calcul raisonnable.



(a) Energies par rapport au paramètre d'échelle.

(b) Temps d'exécution par rapport au nombre de sommets du graphe initial.

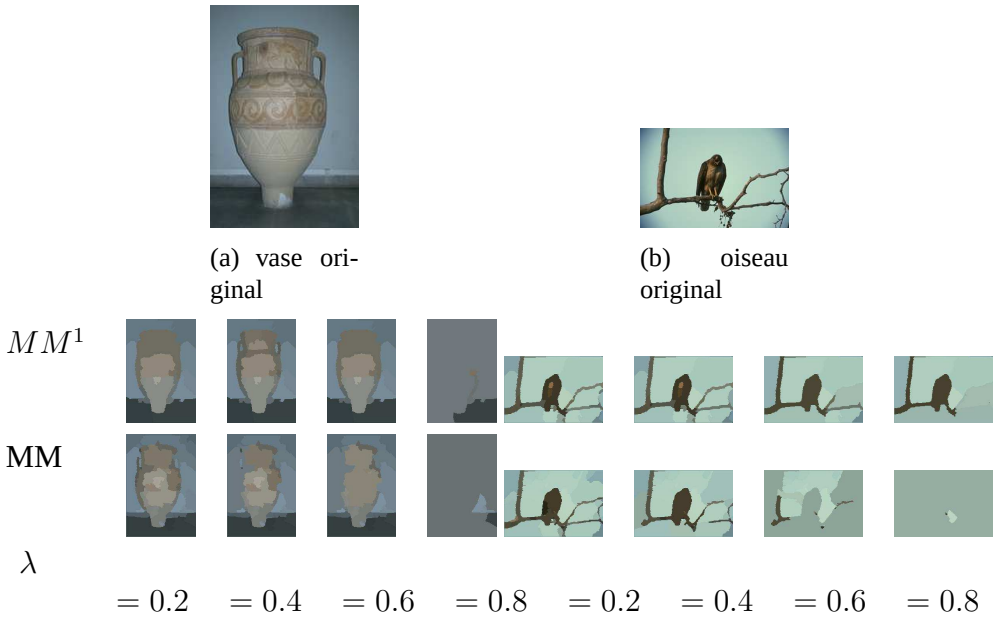
FIG. 3.12 – (a)Energies moyennes des coupes optimales par rapport au paramètre d'échelle pour les algorithmes séquentiels. (b) Temps d'exécution moyen des algorithmes séquentiels par rapport au nombre de régions présentes dans la partition initiale.

3.4.0.b Heuristiques de fusions parallèles

Nous avons dans un second temps comparé les résultats obtenus en utilisant les heuristiques de fusion parallèle (MM) et (MM^1) (voir Section 3.3.2).

La figure 3.13 présente des résultats obtenus en utilisant ces deux heuristiques sur deux images naturelles (Fig. 3.13(a) et (b)). Nous pouvons noter tout d'abord que d'un point de vue visuel, les résultats fournis par (MM^1) semblent meilleurs que ceux de (MM). L'utilisation de l'heuristique (MM^1) permet en effet de préserver des structures assez fines (comme les branches sur les partitions de l'oiseau) même en augmentant le paramètre d'échelle, là ou l'algorithme (MM) les aura déjà fusionnées.

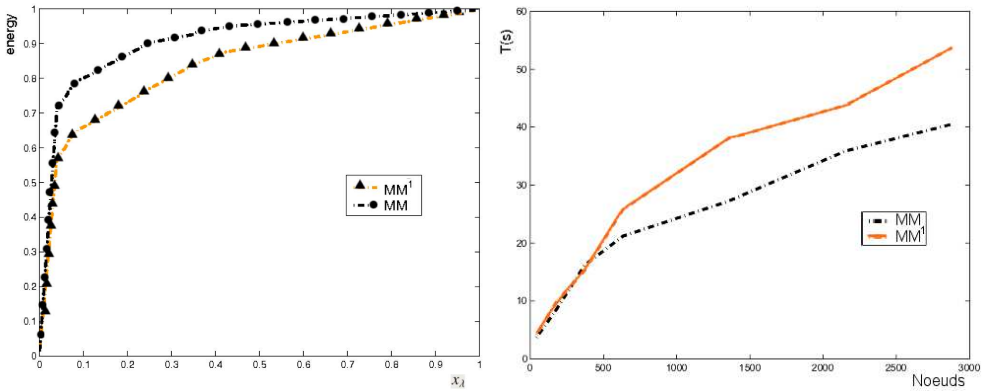
Ces résultat purement visuels sont validés par la figure 3.14(a) qui présente les énergies moyennes des coupes optimales pour chacune de ces heuristiques parallèles. En effet nous pouvons observer que l'énergie moyenne correspondant à l'algorithme (MM^1) est inférieure à celle obtenue avec l'algorithme (MM). La figure 3.14(b) présente pour chacun de ces algorithmes le temps d'exécution moyen nécessaire à la construction de la hiérarchie par rapport au nombre de régions présentes dans la partition initiale. Les temps de construction nécessaires pour les deux algorithmes sont tout à fait comparables avec un léger avantage pour (MM) dès que le nombre de sommets dans la partition initiale dépasse un certain seuil.



(a) vase original

(b) oiseau original

FIG. 3.13 – Partitions des images (a) *vase* et (b) *oiseau* à différentes échelles. La première ligne de ce tableau correspond à l'heuristique MM^1 , la seconde présente les résultats fournis par l'algorithme MM. Les colonnes correspondent aux différentes échelles, le paramètre d'échelle est indiqué en bas pour chacune des colonnes.



(a) Energies par rapport au paramètre d'échelle.

(b) Temps d'exécution par rapport au nombre de sommets du graphe initial.

FIG. 3.14 – (a)Energies moyennes des coupes optimales par rapport au paramètre d'échelle pour les algorithmes parallèles. (b) Temps d'exécution moyen des algorithmes parallèles par rapport au nombre de régions présentes dans la partition initiale.

(a) champignon original

(b) pecheur original

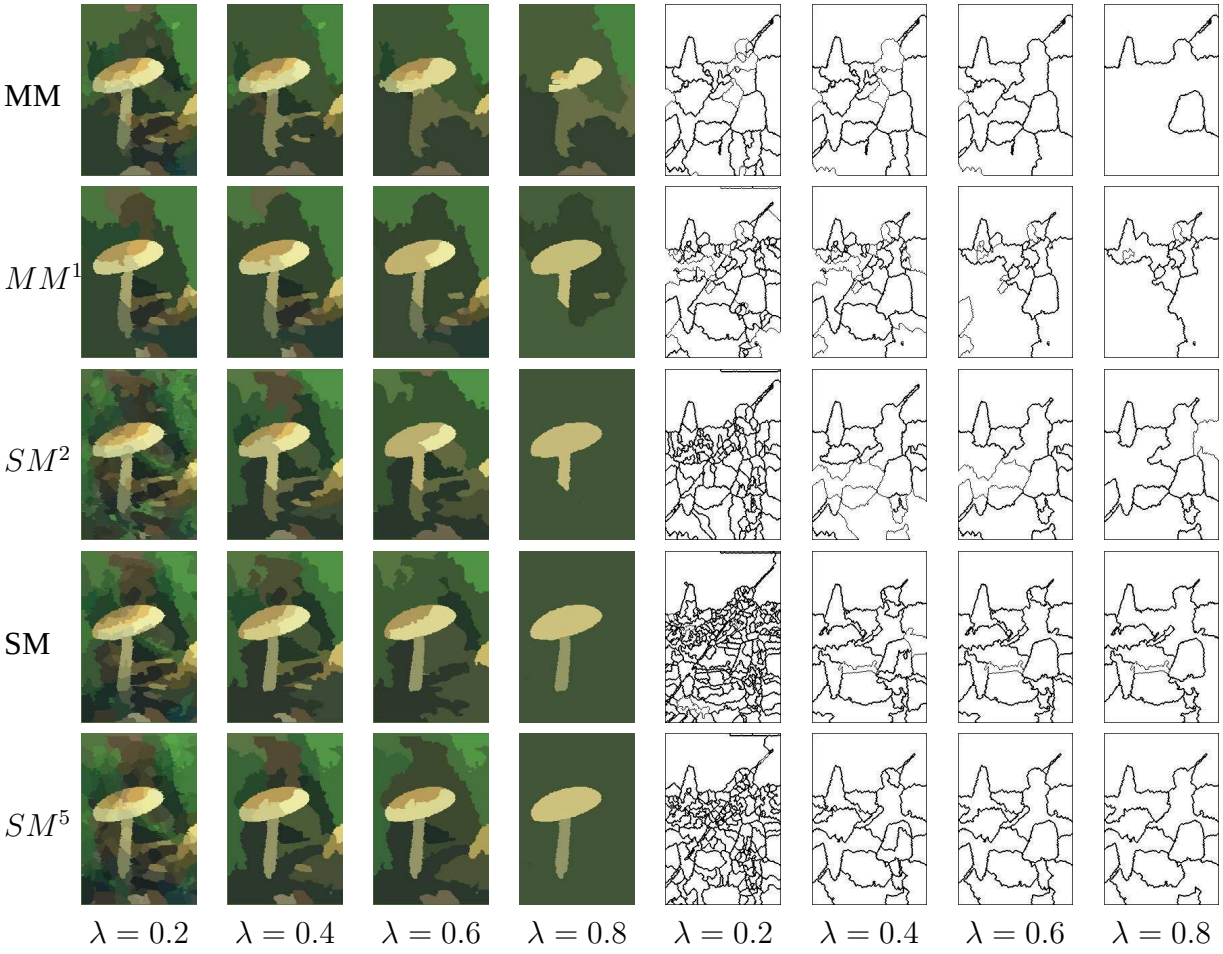
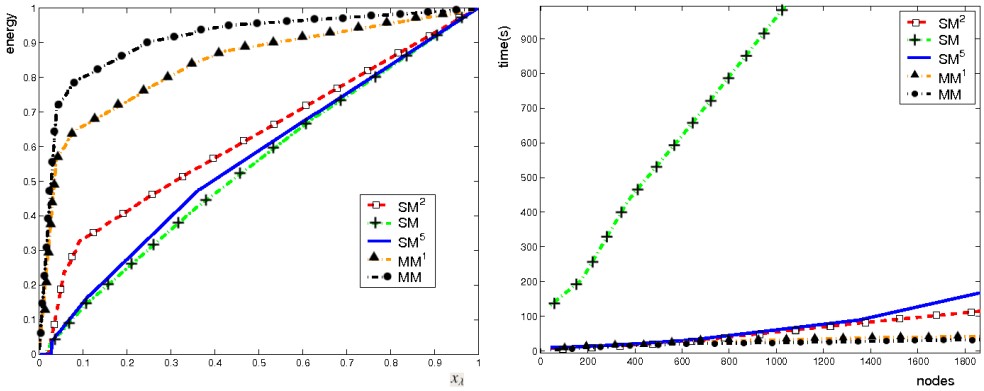


FIG. 3.15 – Partitions des images du *champignon* et du *pêcheur* à différentes échelles. Chaque ligne de ce tableau correspond à une heuristique dont l'acronyme est indiqué dans la première colonne. Les colonnes correspondent aux différentes échelles, le paramètre d'échelle est indiqué en bas pour chacune des colonnes.

3.4.0.c Récapitulatif des différentes heuristiques de fusion.

Nous présentons sur la figure 3.15 les différentes partitions obtenues pour chacune des heuristiques présentées dans la section 3.3, sur les images du *champignon* (Fig. 3.15(a)) et du *pêcheur* (Fig. 3.15(b)).



(a) Energies par rapport au paramètre d'échelle.

(b) Temps d'exécution par rapport au nombres de sommets du graphe initial.

FIG. 3.16 – (a)Energies moyennes des coupes optimales par rapport au paramètre d'échelle pour l'ensemble des algorithmes définis dans la section 3.3 avec (b) les temps d'exécution moyens correspondants.

La figure 3.16 présente un récapitulatif des énergies moyennes, correspondant aux 100 images de la base de Berkeley, produites par l'ensemble des heuristiques présentées avec les temps d'exécution moyens correspondants.

Nous pouvons noter que, malgré le fait que l'énergie des coupes optimales produites par (MM^1) est toujours supérieure à celle de (SM^2), la qualité visuelle des segmentations produites par ces deux algorithmes est relativement équivalente. Le temps d'exécution moyen de (MM^1) est toujours inférieur à celui de (SM^2). Nous pouvons également observer que l'algorithme (MM^1) semble fournir des partitions légèrement plus grossières pour chaque échelle.

Notons que l'ensemble de ces tests ont été effectués sur des machines séquentielles, les heuristiques de fusions parallèles devraient voir leurs temps d'exécution diminuer si nous les implémentions sur des machines permettant une réelle exécution parallèle.

Comme le montre la figure 3.16(a), l'énergie optimale produite par l'heuristique (SM) est toujours inférieure à celle produite par les autres. Il est intéressant de noter que sa courbe caractéristique est très proche de la diagonale du carré unitaire $[0, 1]^2$. Or comme le montre l'équation 3.3 , aucune hiérarchie ne peut obtenir une énergie normalisée inférieure à la diagonale. Ce résultat nous permet de supposer que la solution obtenue par l'algorithme (SM) correspond à l'optimum global dans la majorité des cas. Ce résultat est loin d'être évident a priori puisque (SM) n'explore pas toutes les fusions possibles, mais simplement pour chaque sommet, celles qui se situent autour du sommet initial.

L'heuristique (SM) produit donc des partitions se rapprochant de très près de la solution optimale globale, néanmoins comme le montre la figure 3.16(b) son temps d'exécution est largement supérieur à toutes les autres heuristiques. L'utilisation de l'heuristique

(SM^5), consistant à borner le cardinal de l'ensemble de régions à fusionner à cinq, fournit une énergie très proche de celle de (SM) tout en admettant un temps d'exécution largement inférieur. L'heuristique (SM^5) semble apparaître comme un excellent compromis permettant d'obtenir des énergies relativement faibles tout en conservant un temps de calcul raisonnable.

Si le temps d'exécution est le facteur à privilégier, l'heuristique (MM^1), semble apparaître comme un excellent compromis entre le temps de calcul et la qualité visuelle générale des segmentations produites.

3.4.0.d Influence de la partition initiale

En dehors de l'énergie E_λ, les algorithmes présentés dépendent uniquement de la sur-segmentation initiale. Nous avons considéré deux types de sur-segmentation pour mesurer l'influence de la partition initiale sur les résultats. La sur-segmentation triviale correspondant à la grille de pixels de l'image initiale et la sur-segmentation obtenue par un algorithme de ligne de partage des eaux (LPE).

La figure 3.17 présente des résultats de segmentation hiérarchique obtenus à partir de l'algorithme (SM^2) en partant d'une partition initiale différente. La première ligne, correspondant aux partitions obtenues en partant de la grille de pixels initiale, donne des résultats extrêmement proches de ceux obtenus sur la seconde ligne, qui correspond aux partitions obtenues en partant d'une sur-segmentation par ligne de partage des eaux. Les résultats sont naturellement assez différents pour des petites valeurs du paramètre d'échelle, mais l'on voit que les coupes réalisées à grande échelle sont relativement similaires.

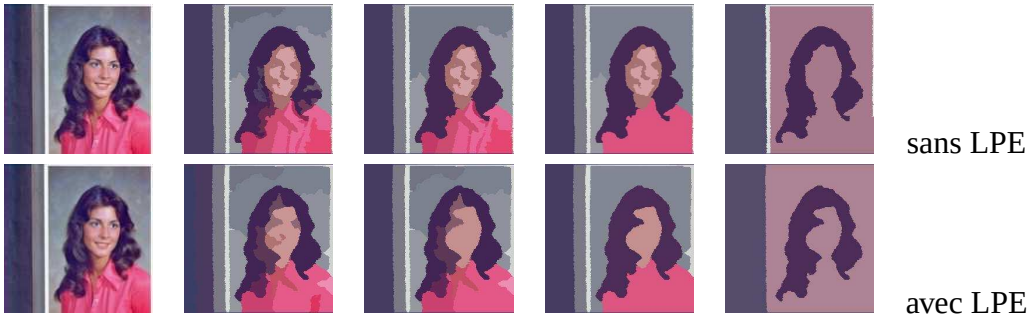
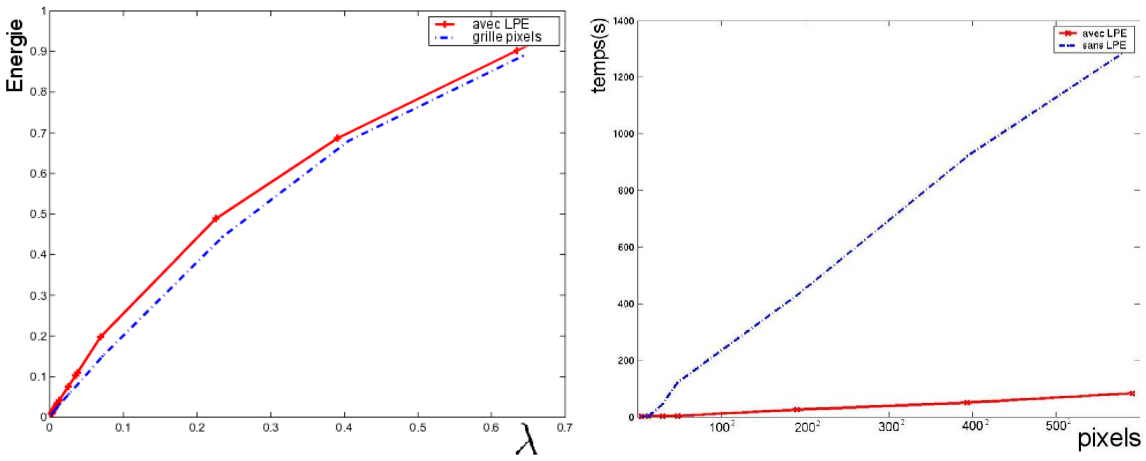


FIG. 3.17 – Comparaison de résultats de segmentation obtenus avec une partition initiale différente. La première ligne correspond aux partitions obtenues en partant de la grille initiale, chaque pixel représentant une région. La seconde ligne correspond aux partitions obtenues en partant d'une sur-segmentation par ligne de partage des eaux. Les colonnes correspondent aux différentes échelles, le paramètre d'échelle est indiqué en bas pour chacune des colonnes.

La figure 3.18(a) présente ces résultats d'un point de vue énergétique alors que la figure 3.18(b) permet de comparer le temps d'exécution nécessaire à la construction de la hiérarchie H pour chacune des partitions initiales. Même si l'énergie obtenue en partant de la grille des pixels est légèrement inférieure à celle obtenue en effectuant une ligne de partage des eaux, nous pouvons noter que les deux énergies sont extrêmement proches.

Le temps d'exécution nécessaire est, quant à lui, beaucoup plus important en partant de la grille des pixels.



(a) $E_\lambda(C^*_\lambda(H))$ (b) temps d'exécution

FIG. 3.18 – (a) Comparaison de l'énergie $E_\lambda(C^*_\lambda(H))$ obtenue avec une partition initiale différente (b)ainsi que des temps d'exécution en fonction de la taille de l'image, nécessaires à la construction de la hiérarchie H

L'algorithme (SM^2) est donc remarquablement stable vis a vis de la partition initiale. L'utilisation d'un algorithme de ligne de partage des eaux s'avère être très intéressant si l'on ne s'intéresse pas spécialement aux micro-structures de l'image. Il permet en effet de réduire considérablement le temps d'exécution tout en fournissant des résultats proches, aussi bien qualitativement que quantitativement, de ceux obtenus en partant de la grille des pixels.

3.4.0.e Energie contraste Min/Max

Les algorithmes présentés dans cette section, ont tous été testés avec l'énergie de Mumford & Shah [MUMFOR89] (i.e équation 3.13).

Nous avons également étudié leur comportement en utilisant une énergie présentant un terme d'attache aux données différent, basé sur la notion de contraste interne/externe. L'idée principale de ce critère [FELZEN98] est basée sur le principe qu'une région doit présenter un contraste plus important avec ses voisins qu'avec ses éventuelles sous-parties. Nous pouvons ainsi définir la notion de *contraste externe* (avec ses voisins) et de *contraste interne* (avec ses sous-parties) pour toute région.

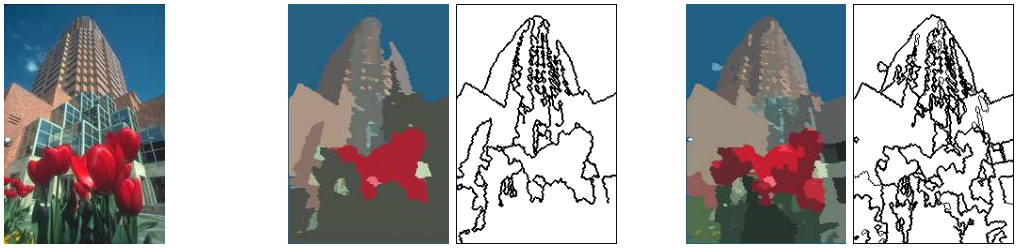
Soit G_e le gradient moyen calculé le long de la portion de contour associée à une arête e. Le contraste interne d'une région R est défini par

$$Int(R) = max_{e \in CC(R)} G_e$$

et le contraste externe est défini par

$$Ext(R) = min_{e \in E | v \in \iota(e)} G_e$$

$CC(R)$ représente l'ensemble des arêtes qui ont été contractées pour obtenir la région R et $e \in E | v \in \iota(e)$ représente l'ensemble des arêtes incidentes à v.



(a) Image originale (b) $D(R) = SE(R)$ (c) $D(R) = SE(R)(1 + f(\frac{Int(R)}{Ext(R)})$

FIG. 3.19 – (a) Image originale. (b) et (c) partitions de l'image de la tour construites à partir de la même heuristique (SM) pour le même paramètre d'échelle ($x_\lambda = .6$) mais avec des énergies utilisant un terme d'attache aux données différent. (b) utilise l'erreur quadratique $D(R) = SE(R)$ tandis que (c) est définie en utilisant la formule définie par l'équation 3.14

La nouvelle énergie que nous utilisons peut donc être interprétée comme une combinaison de la notion de contraste interne/externe utilisée par Haxhimusa [HAXHIM03], et de l'erreur quadratique. Elle se définit de la façon suivante :

$$E_\lambda(P) = \sum_{i=1}^{n} SE(R_i)\left(1 + f\left(\frac{Int(R_i)}{Ext(R_i)}\right)\right) + \lambda|\delta(R_i)| \tag{3.14}$$

avec $f()$ représentant une fonction sigmoïde.

Les régions fortement contrastées présentent ainsi un faible rapport entre les contrastes interne et externe. Inversement, une région faiblement contrastée présentera un terme d'attache aux données proche de deux fois son erreur quadratique. Si l'on se réfère à l'équation 3.10, la création d'une région faiblement contrastée sera associée à une grande échelle d'apparition et sera donc pénalisée.

La figure 3.19 présente une comparaison de résultats obtenus par la même heuristique (SM), mais en utilisant deux termes d'attache aux données différents sur l'image de la tour. La figure 3.19(b) présente ainsi les résulats de (SM) en utilisant l'erreur quadratique $D(R) = SE(R)$ comme attache aux données alors que l'énergie utilisée pour obtenir figure 3.19(c) est définie en utilisant $D(R) = SE(R)(1+f(\frac{Int(R)}{Ext(R)})$ comme terme d'attache aux données. On peut remarquer que les zones contrastées sont mieux conservées avec ce nouveau critère de fusion. Dans la figure 3.19, le nuage présentant un fort contraste, situé à gauche de la tour, fusionne avec le ciel sur la figure 3.19(b) alors qu'il est toujours présent à la même échelle sur la figure 3.19(c) qui utilise cette nouvelle énergie comme terme d'attache aux données.

DEUXIÈME PARTIE

APPARIEMENT HIÉRARCHIQUE

MÉTHODES D'APPARIEMENT

Sommaire

LA mise en correspondance de structures est une étape préliminaire à de nombreux traitements. Dans le domaine du traitement d'images, les méthodes d'appariement structurel sont en général effectuées sur des points d'intérêts ou une représentation de l'image en régions. Les relations entre ces objets fournissent des graphes et le problème peut alors être résolu par la mise en correspondance des graphes représentatifs de ces images. Il se transforme ainsi en un problème d'appariement de graphes.

L'appariement de graphes est un problème relativement ancien qui regroupe quatre problèmatiques assez proches :

1. l'isomorphisme de graphes : Permet de vérifier si deux graphes sont identiques à une renumérotation des sommets et des arêtes prêt,
2. l'isomorphisme de sous-graphes : Permet de déterminer s'il existe un isomorphisme entre un graphe et le sous-graphe d'un autre graphe,
3. le plus grand sous-graphe commun : Permet de déterminer un isomorphisme maximum entre deux sous-graphes,
4. distance d'édition entre graphes : fournit le nombre minimum de transformations à appliquer à deux graphes pour qu'ils deviennent isomorphes.

Dans le domaine qui nous intéresse, à savoir le traitement d'images, divers procédés vont permettre d'obtenir une représentation naturelle de nos données par des graphes valués.

En effet divers procédés tels qu'une squelettisation, une segmentation, ou une détection de primitives géométriques (points d'intérêts, lignes, polygones . . .) fournissent différents types de graphes. Dans le cadre du traitement d'image, les noeuds et les arêtes des graphes manipulés possèdent généralement une interprétation aussi bien géométrique que sémantique, fournissant des distances entre ces noeuds ou ces arêtes. Les graphes manipulés peuvent couramment compter plusieurs milliers de noeuds, ce qui peut amener à des temps de calculs importants, incompatibles avec des contraintes temps réel, nécessaires dans certaines applications de ce domaine.

Notre objectif est de déterminer les similarités entre des images ou des formes à partir de leur représentation par graphes. Même si les premiers travaux dans ce domaine sont anciens, leur application à des données de grande taille demeure relativement récente.

Les domaines d'applications les plus remarquables pour l'isomorphisme, le morphisme et le plus grand sous graphe commun sont (i) l'indexation d'images [MULHEM01] (ii) la reconnaissance de formes ou d'objets [ABDULR98] et (iii) le suivi de segmentations dans les vidéos [CONTE06]. Pour tous les algorithmes permettant des calculs de distances, les domaines privilégiés sont (i) l'indexation d'images [MULHEM01] et (ii) la classification de formes [FOGGIA07].

Après quelques définitions (section 4.1) nous étudierons deux grandes familles de méthodes utilisées pour l'appariement de graphes, les approches algorithmiques (sections 4.2) puis celles par optimisation (section 4.3). L'appariement de partitions sera ensuite étudié en section 4.4.

4.1 Appariement de graphes

4.1.1 Définitions

Nous allons tout d'abord définir les notions de base indispensables à la poursuite de cette étude.

Définition 70 Isomorphisme de graphes

*Étant donnés deux graphes $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$, un **isomorphisme** de graphes entre G_1 et G_2 est une bijection φ entre l'ensemble des sommets V_1 de G_1 et l'ensemble de sommets V_2 de G_2 telle que :*

$$(v_1, v_2) \in E_1 \Leftrightarrow (\varphi(v_1), \varphi(v_2)) \in E_2$$

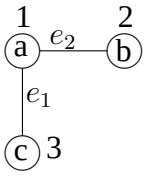
*S'il existe un isomorphisme entre deux graphes, ces deux graphes sont dits **isomorphes**. Si la bijection est un appariement d'un graphe sur lui même (i.e. si $G_1 = G_2$), cette bijection est appelée **automorphisme**.*

Définition 71 Sous graphe commun

Étant donnés deux graphes G_1 et G_2, G est un sous graphe commun de G_1 et G_2 si et seulement si il existe :
$G \xrightarrow{\phi} G_1$
$G \xrightarrow{\psi} G_2$
avec ϕ et ψ isomorphismes de sous-graphe

*G sera de plus le sous graphe **maximum** si l'on ne peut pas trouver de sous graphe de cardinal supérieur.*

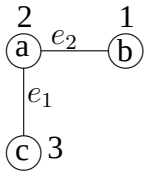
Définition 72 Appariement inexact de graphes



(a) G_M

$$\begin{pmatrix} a & e_2 & e_1 \\ e_2 & b & 0 \\ e_1 & 0 & c \end{pmatrix}$$

(b) M



(c) $G_{M'}$

$$\begin{pmatrix} b & e_2 & e_1 \\ e_2 & a & 0 \\ e_1 & 0 & c \end{pmatrix}$$

(d) M'

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

(e) P

FIG. 4.1 – (a) Un graphe labellisé G_M avec sa matrice d'adjacence M (b) et le graphe $G_{M'}$ (c) accompagné de sa matrice d'adjacence M' (d) obtenue par application d'une matrice de permutation P (e)

Un appariement inexact de deux graphes G_1 et G_2 est défini par un isomorphisme entre deux graphes, obtenus à partir de G_1 et G_2 en effectuant des opérations d'édition telles que des substitutions ou des suppressions de sommets ou d'arêtes.

*Cet appariement est dit **optimal** si le coût relatif à la séquence d'opérations d'édition nécessaires pour rendre les graphes isomorphes est minimale.*

Définition 73 Matrice d'adjacence *La **matrice d'adjacence** $M = (m_{i,j})$ d'un graphe labellisé $G = (V, E, \mu, \nu, L_v, L_e)$ (voir Déf. 9) est une matrice carrée dont tous les éléments non-diagonaux $m_{i,j}$ avec $i \neq j$ indiquent la présence ou non d'une arête entre le sommet i et le sommet j, alors que les éléments diagonaux $m_{i,i}$ portent le label du sommet i :*

$\forall i \in \{1, \ldots, n\}\ m_{i,i} = \mu(v_i)$

$\forall (i,j) \in \{1, \ldots, n\}^2, i \neq j$

$$m_{i,j} = \begin{cases} \nu((v_i, v_j)) & \text{si } (v_i, v_j) \in E \\ 0 & \text{sinon} \end{cases}$$

Pour un graphe non labellisé $G = (V, E)$ les éléments de la matrice seront à 0 ou 1 selon la présence ou non d'une arête entre les sommets considérés

$\forall (i,j) \in \{1, \ldots, n\}^2, i \neq j$

$$m_{i,j} = \begin{cases} 1 & \text{si } (v_i, v_j) \in E \quad \text{ou si } i = j \\ 0 & \text{sinon} \end{cases}$$

Définition 74 Matrice de permutation *Une **matrice de permutation** $P = (p_{i,j})$ est une matrice carrée qui vérifie les propriétés suivantes :*

1. *tous ses coefficients sont à 0 ou 1 :*
   $\forall (i,j) \in \{1, \ldots, n\}^2\ p_{i,j} \in \{0, 1\}$,
2. *il y a exactement un 1 par ligne et par colonne*
   $\forall j \in \{1, \ldots, n\}\ \sum_{i=0}^{n} p_{i,j} = 1, \forall i \in \{1, \ldots, n\}\ \sum_{j=0}^{n} p_{i,j} = 1.$

Nous pouvons illustrer ces notions grâce à la figure 4.1(a) représentant un graphe labellisé ayant comme matrice d'adjacence la matrice M présentée sur la figure 4.1(b).

Si l'on applique à la matrice M la matrice de permutation P suivante :

$$P = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

nous obtenons la matrice d'adjacence M' (voir figure 4.1(d)) représentant le graphe présenté sur la figure 4.1(c).

Le problème d'isomorphisme de graphes peut ainsi être redéfini en termes de matrice. Deux graphes G_1 et G_2, de matrices M_1 et M_2, sont dits **isomorphes** si et seulement si il existe une matrice de permutation P telle que :

$$M_2 = PM_1P^t$$

On vérifie facilement que cette définition est compatible avec la définition de l'isomorphisme de graphes (voir Déf. 70)

Les matrices d'adjacence et de permutation ont été les outils de base permettant à Ullman de définir en 1976 un des tous premiers algorithmes d'isomorphisme de graphes [ULLMAN76]. Le principal problème de cet algorithme demeure sa très forte complexité. Il lui faut en effet effectuer autant de tests d'isomorphisme, que le graphe compte de sommets, pour aboutir. Ceci engendre un nombre important de calculs et limite son usage à des graphes de très petite taille.

Définition 75 Clique

Soit $G = (V, E)$ un graphe non orienté, une clique dans G est un sous-ensemble $V_c \subset V$ de sommets de V, tel que pour tout couple de sommets $(v_1, v_2) \in V_c \times V_c$, il existe une arête $e \in E$ reliant v_1 à v_2. On dira également d'un graphe dont tous les sommets sont connectés les uns aux autres qu'il définit une clique.
La taille de la clique K, généralement notée $|K|$, est définie par le nombre de sommets qu'elle comporte.

*Une clique est dite **maximale** dans G si elle n'est contenue dans aucune autre clique de taille supérieure et **maximum** si elle définit la plus grande clique de G. La clique maximum d'un graphe G est notée $\omega(G)$.*

L'isomorphisme de sous-graphe a également été étudié par le biais d'une structure appelée *graphe d'association*. L'idée consiste à construire un graphe d'association et d'en calculer les cliques qui nous donneront l'ensemble des isomorphisme de sous graphes.

Définition 76 Graphe d'association

Soient $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$, le graphe d'association $G = (V, E)$ de G_1 et G_2 est défini tel que

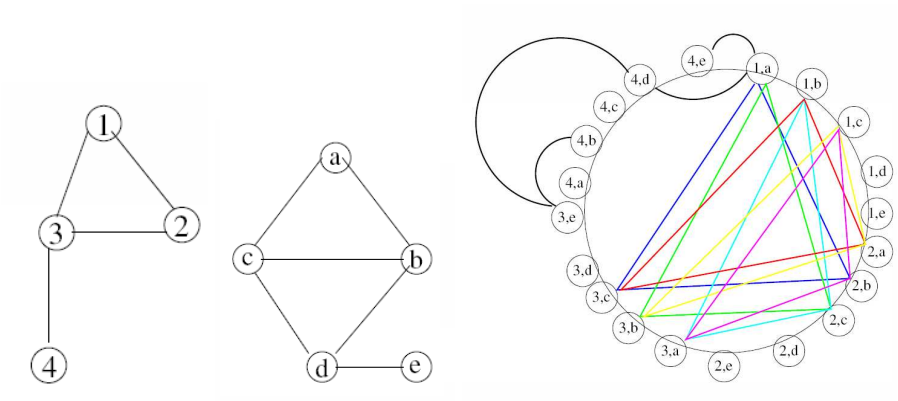
- *Ses sommets sont issus du produit cartésien des sommets de G_1 et G_2 :*

$$V = V_1 \times V_2$$

- *Ses arêtes sont définies par :*

$$E = \{((i,h),(j,k)) \in V \times V | i \neq j, h \neq k \text{ et } (i,j) \in E_1 \Leftrightarrow (h,k) \in E_2\}$$

Les arêtes du graphe d'association (qui ne sont pas dirigées) représentent la potentialité qu'ont les paires de sommets, d'appartenir à l'appariement. Un nœud correspondant à la paire (n_1, n_2) est connecté à un nœud (m_1, m_2) si et seulement si le fait d'apparier les sommets n_1 et n_2, n'interdit pas l'appariement de m_1 avec m_2, et inversement. La figure 4.2 illustre cette définition en présentant le graphe d'association (Fig 4.2(c)) , construit à partir de deux graphes G_1 et G_2 (Fig 4.2(a) et (b)).



(a) G_1 (b) G_2 (c) graphe d'association

FIG. 4.2 – Exemple de cliques dans un graphe d'association issues du couple de graphes (G_1, G_2).

Calculer les cliques du graphe d'association revient alors à calculer les sous graphes communs de G_1 et G_2.

4.1.2 Distance d'édition entre graphes

Contrairement aux méthodes par optimisation, fournissant parfois des énergies facilement comparables, les graphes n'offrent pas directement de modèle de distance telle que la distance euclidienne. Une approche classique pour définir une mesure de dissimilarité entre deux graphes consiste à déterminer le nombre minimum de transformations nécessaires, pour passer d'un graphe à l'autre. Les transformations utilisées sont généralement des suppressions, insertions ou des substitutions de nœuds ou d'arêtes.

Définition 77 *La distance entre deux graphes $G = (V, E)$ et $G' = (V', E')$ peut être définie en termes d'opérations d'édition élémentaires nécessaires à la transformation du graphe G en G' avec un coût minimum. Les trois opérations d'édition de base sont :*

1. ***substitution** d'un sommet $x \in V$ par un sommet $y \in V'$, notée $x \rightarrow y$*
2. ***insertion** d'un sommet $x \in V$, notée $\lambda \rightarrow x$*
3. ***suppression** d'un sommet $x \in V$, notée $x \rightarrow \lambda$*

dans lesquelles λ représente le sommet vide.

Définition 78 Séquence d'édition

*Une **séquence d'édition** S est définie comme une séquence ordonnée d'opérations d'édition élémentaires $s_1, \ldots, s_p$.*

Pour deux graphes structurellement proches, il existe ainsi une séquence d'édition de faible coût alors que pour deux graphes dont la structure est très différente une séquence d'édition, dont le coût est important, sera nécessaire.

La **distance d'édition** de deux graphes est alors définie comme le chemin d'édition de coût minimum entre ces deux graphes.

Définition 79 Distance d'édition

Soit c une fonction de coût qui attribue à chaque opération d'édition s une valeur réelle positive $c(s)$. Le coût d'une séquence d'édition est défini comme la somme des coûts de chacune des opérations d'édition élémentaires qui la composent :

$$c(S) = \sum_{i=1}^{p} c(s_i) \tag{4.1}$$

*La **distance d'édition** entre deux graphes G et G' est alors définie comme le coût minimum correspondant à une séquence d'édition permettant de transformer G en G' :*

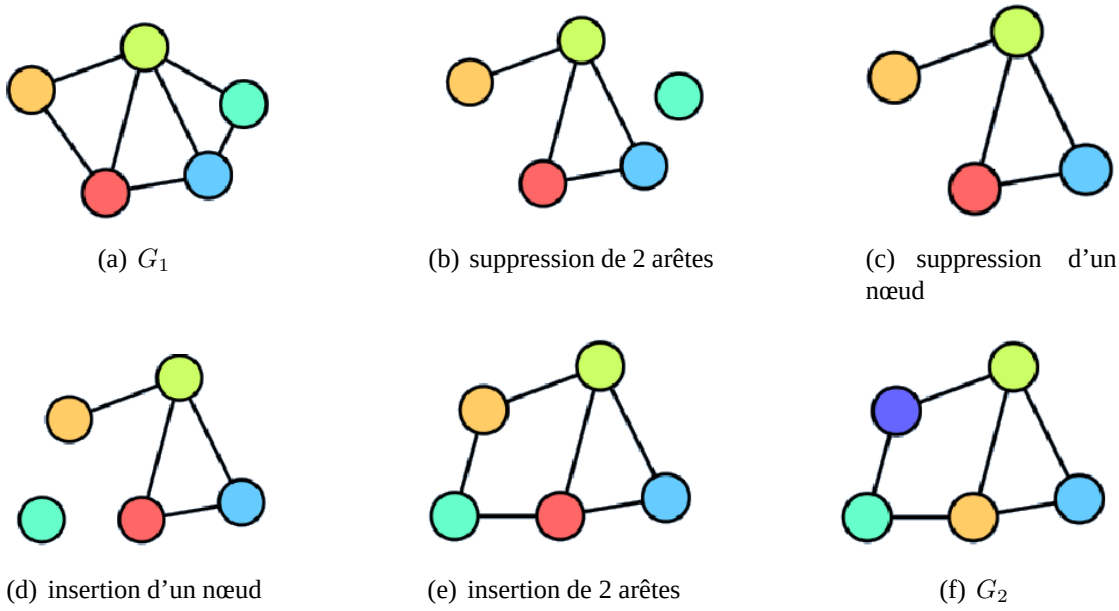
$$d(G, G') = \min\{c(S) : \textit{S est une séquence d'opérations d'édition qui transforme } G \textit{ en } G'\} \tag{4.2}$$

La figure 4.3 présente les opérations d'édition nécessaires pour passer du graphe G_1 (Fig. 4.3(a)) au graphe G_2 (Fig. 4.3(f)). Dans cet exemple, le chemin d'édition est composé de deux suppressions d'arêtes, suivi de la suppression d'un nœud. Il faut ensuite insérer un nouveau nœud, puis deux nouvelles arêtes et l'on obtient finalement le graphe G_2 après une relabellisation de certain nœuds (opération de substitution).

La distance d'édition est généralement calculée en utilisant des files de priorité ou par le biais d'un algorithme basé sur un arbre de recherche, dans lequel les chemins d'édition possibles sont itérativement explorés, permettant ainsi de retrouver le chemin d'édition minimum [BUNKE83]. Le principal inconvénient de ces approches réside dans le fait que la complexité en temps est exponentielle, ce qui contraint son utilisation à des graphes de petite taille. Des heuristiques sous-optimales ont néanmoins été proposées permettant d'accélérer le calcul, rendant ce type d'algorithmes utilisable sur des graphes comportant un nombre de nœuds plus important.

4.1.3 Complexité de l'appariement de graphes

Le problème d'appariement de graphes et de sous graphes est un problème qui a été énormément étudié [CORNEI70, HOPCRO72, BERZTI73, ULLMAN76, GHAHRA80, BUNKE95, MESSME98, CORDEL04]. Il est considéré comme un des problèmes les plus complexes dans le domaine de la reconnaissance de formes en vision par ordinateur [BIENEN87]. Sa complexité est principalement dûe à son aspect fortement combinatoire.



(a) G_1 (b) suppression de 2 arêtes (c) suppression d'un nœud

(d) insertion d'un nœud (e) insertion de 2 arêtes (f) G_2

FIG. 4.3 – Différentes opérations d'édition nécessaires pour passer du graphe G_1 au graphe G_2.

4.1.3.a Appariement exact de graphes : isomorphisme de graphe

Un algorithme d'appariement de graphes est dit *optimal* s'il permet de trouver toutes les solutions. Un tel algorithme est donc capable de trouver l'ensemble des appariements possibles entre deux graphes.

Toutes les catégories d'appariement de graphes n'ont pas encore été affectées à une classe particulière de complexité comme P ou NP-complet. De nombreux travaux tentent de prouver l'aspect NP-complet quand les deux graphes à apparier sont dans une configuration particulière ou satisfont à certaines contraintes [BASIN94, GAREY90] mais la démonstration prouvant que le problème général de l'isomorphisme de graphe est bien NP-complet n'a toujours pas été donnée.

4.1.3.b Appariement exact de sous-graphes : isomorphisme de sous-graphe

Garey et Johnson [GAREY90] ont prouvé que ce type particulier de problème d'appariement de graphes est NP-complet. Néanmoins, pour certains types de graphes il est possible d'obtenir une complexité inférieure. Par exemple, le cas particulier dans lequel le plus gros graphe est une forêt et le plus petit des deux graphes à apparier est un arbre, a été démontré comme étant de complexité polynomiale [REYNER77, GAREY90].

4.1.3.c Appariement inexact de graphes

Le problème consiste à trouver un coût minimal pour assigner les nœuds du graphe G_1 à ceux du graphe G_2. Pour deux graphes ayant respectivement n et m nœuds il y a

théoriquement $n(n-1)\dots(n-m+1)$ appariements possibles parmi lesquels certains sont optimum. La complexité en temps d'un algorithme naïf est en $\mathcal{O}(n^n)$. L'algorithme de Munkres [MUNKRE57] résoud ce problème, dans le cas de graphes bipartites en $\mathcal{O}(n^3)$ avec n le nombre maximum de nœud des deux graphes.

4.1.3.d Appariement inexact de sous-graphes : homomorphisme de graphe/sous-graphe

Il a été démontré par Abdulkader [ABDULK98] que le cas de l'appariement inexact de graphes est un problème NP-complet. Il a été montré de manière similaire que la complexité du problème d'appariement inexact de sous-graphes admet une complexité équivalente à celle du problème du plus grand sous-graphe commun, qui est connu comme étant également NP-complet.

4.1.3.e Bilan

Les problèmes touchant à l'appariement de graphes demeurent donc des problèmes *difficiles* nécessitant la mise en place de fortes heuristiques si l'on espère obtenir, en un temps raisonnable, une bonne solution au problème posé. Le problème d'isomorphisme de graphes a été abordé de plusieurs points de vue et les deux approches les plus abouties sont les approches par optimisation, dites numériques, et les approches purement algorithmiques.

4.2 Approche algorithmique

Les méthodes algorithmiques permettent de calculer l'isomorphisme de graphe en effectuant une construction explicite de l'appariement et permettent de fournir une solution optimale au problème.

4.2.1 Approche par représentation de l'espace des états

L'approche par représentation de l'espace des états, généralement abrégée en SSR (pour *State Space Representation*), consiste à définir un algorithme capable d'effectuer une recherche dans l'espace des solutions, en rejetant a priori certaines solutions.

Pour résoudre le problème du plus grand sous graphe commun, McGregor propose un premier algorithme [MCGREG82], pouvant être décrit par une représentation de l'espace d'état (SSR). Partant d'un état initial vide un couple de noeuds, appartenant à chacun des deux graphes à apparier et n'ayant pas encore été testés, est sélectionné à chaque étape. Il faut ensuite vérifier si le plus grand sous graphe commun courant peut être étendu par l'ajout de cette paire de nœuds. Si cette extension est possible, la solution partielle courante est étendue par l'ajout du couple (n_1, n_2). Si cette solution donne un sous graphe plus important que la meilleure des solutions obtenues jusqu'à présent, cette nouvelle solution est conservée.

Après cet ajout si l'état courant s n'est pas une feuille (Déf. 5) dans l'arbre de recherche (Déf. 4), un nouvel état est généré et son analyse est effectuée. Une procédure de *backtrack* est utilisée pour tester les différentes extensions possibles. Les recherches le long d'une branche se terminent lorsqu'une feuille est atteinte ou lorsque le nombre de nœuds, de l'état courant jusqu'à la feuille de la branche analysée, ne suffit pas pour construire un sous graphe plus grand que celui qui est actuellement stocké.

Un inconvénient de cette méthode est que toutes les branches de l'arbre de recherche doivent être explorées, car il n'est pas possible de savoir à l'avance si une branche contiendra une meilleure solution que celle actuellement stockée. Cet algorithme ne nécessite de stocker que l'état de la branche contenant le nœud solution. Pour deux graphes possédant respectivement N_1 et N_2 sommets, avec $N_1 \leq N_2$, chaque branche de l'arbre de recherche ne peut avoir une profondeur supérieure à N_1. La complexité en mémoire de cet algorithme est donc constante, en $\mathcal{O}(N_1)$.

4.2.2 Problème du sous-graphe commun approché par la méthode des cliques

Pour résoudre le problème du sous-graphe commun Durand et Parisi [DURAND99] proposent en 1999 un algorithme basé sur la notion de clique (Déf. 75). La première étape consiste à construire le graphe d'association (Déf. 76) dont les sommets correspondent aux paires de sommets des deux graphes à apparier. L'algorithme génère ensuite une liste de sommets, représentant une clique dans le graphe d'association, en utilisant une stratégie de recherche en profondeur dans un arbre de recherche. Il sélectionne systématiquement tous les nœuds des niveaux successifs jusqu'à ce qu'il ne soit plus possible d'ajouter de sommets à la liste.

Algorithme 1 Algorithme de DurandParisi

```
entrée : un graphe d'association
sortie : la clique maximale
procédure DurandParisi(s)
while prochainNœud(s,n) do
  if estLegal(s,n) then
    s :=AjouterNœud(s,n)
    if taille(s') > tailleMax then
      sauverClique(s') ; tailleMax=taille(s')
    end if
    if non feuille(s') then
      DurandParisi(s')
    end if
    Backtrack(s')
  end if
end while
```

L'algorithme 1 présente en détail le déroulement de cet algorithme. La première étape consiste à vérifier pour chaque nœud visité s'il est *legal* (i.e s'il est connecté à tous les autres nœuds de la clique). Si c'est le cas, l'algorithme vérifie si la taille de la clique

obtenue en ajoutant ce sommet est plus grande que celle de la plus grande clique stockée jusqu'à présent. Dans ce cas la plus grande clique ainsi que sa nouvelle taille sont mises à jour.

La complexité mémoire de cet algorithme est déterminée par le stockage du graphe d'association qui, pour deux graphes possédant respectivement N_1 et N_2 sommets (avec $N_1 \leq N_2$) contient $N_1.N_2$ sommets et au pire $(N_1.N_2)^2$ arêtes.

Ce type d'algorithme a l'avantage d'être peu calculatoire mais présente l'inconvénient d'être extrêmement gourmand en espace mémoire, du fait du stockage du graphe d'association qui peut être de taille conséquente.

Cet algorithme basé sur le calcul de la clique maximale a été comparé sur différents types de graphes à celui de SSR de McGregor (i.e section 4.2.1). Il en ressort qu'aucun d'entre eux n'est meilleur que l'autre dans tous les cas de figure. Du fait qu'il ne nécessite pas la construction ni le stockage du graphe d'association, l'algorithme de McGregor s'avère être le plus rapide dans le cas d'utilisation avec des graphes de faible densité et de petite taille. En revanche pour des graphes plus denses, les pré-calculs effectués par l'algorithme de Durand-Parisi se révèlent être plus efficaces.

4.2.2.a Algorithmes SM^i

Fosser et al ont plus récemment proposé plusieurs heuristiques combinatoires [FOSSER03], appelées SM^i avec $i = 1, 2$ ainsi qu'une évolution notée SM_SWAP permettant de résoudre le problème de la clique maximale.

L'algorithme SM^i (voir Algorithme 2) est basé sur un pré-calcul de toutes les cliques de taille i. Pour chaque clique initiale K, il tente d'agrandir celle-ci en ajoutant des sommets ne faisant pas partie de K et étant reliés par une arête à chaque sommet de K. L'ensemble des sommets candidats vérifie donc l'équation :

$$C_0(K) = \{j \in V - K \,|\, \forall k \in K \,(j,k) \in E\} = \bigcap_{k \in K} N_k \tag{4.3}$$

avec N_j représentant le voisinage d'un sommet j :

$$N_j = \{k \in V \,|\, (j,k) \in E\}$$

Pour maximiser les chances d'obtenir une clique de taille maximale, l'algorithme SM^i sélectionne à chaque étape le sommet de $C_0(K)$ qui a le plus de voisins dans $C_0(K)$:

$$l := arg \max_{j \in C_0(K)} |C_0(K) \cap N_j| \tag{4.4}$$

Expérimentalement l'algorithme SM^1, basé sur les sommets, donne des résultats satisfaisants avec un temps d'exécution très bas. L'algorithme SM^2, basé sur les couples de sommets donne de meilleurs résultats mais admet un temps d'exécution largement supérieur [M. LOC02].

Alors que l'algorithme SM^i part d'une clique de taille i et ajoute de nouveaux sommets à chaque itération, jusqu'à ce qu'une clique maximale soit atteinte, l'algorithme

Algorithme 2 Algorithme SM^i

entrée : un graphe $G = (V, E)$
sortie : la clique maximale K
Q := pile contenant toutes les cliques de cardinal i
$\mathcal{K} := \emptyset$
$K^* := \emptyset$
$max := 0$
while Q $\neq \emptyset$ **do**
  $H := pop(Q)$
  $K := H$
  while $C_0(K) \neq \emptyset$ **do**
    $l := arg\max_{j \in C_0(K)} |C_0(K) \cap N_j|$
    $K := K \cup \{l\}$
  end while
  $K := K \cup \{l\}$
  if $|K| > max$ **then**
    $max := |K|$
    $K^* := K$
  end if
end while
renvoyer K^*

SM_SWAP propose de ne pas systématiquement ajouter un nouveau sommet mais également d'intervertir certains sommets. Cet échange de sommets a pour but d'éviter la convergence vers le minimum local correspondant à la clique de départ, permettant ainsi d'atteindre ou de s'approcher du maximum global.

L'algorithme SM_SWAP utilise le même ensemble de sommets candidats à l'adjonction $C_0(K)$ (équation 4.3) et définit en plus un ensemble $C_1(K)$ de sommets candidats à l'échange. Cet ensemble représente alors les sommets n'appartenant pas à K et étant connectés à tous les sommets de K excepté un :

$$C_1(K) = \{j \in V - K \quad | \quad N_j \cap K = |K| - 1\} \tag{4.5}$$

Un sommet j appartient alors à l'ensemble $C_1(K)$ si et seulement si un seul sommet de la clique courante K lui interdit d'entrer dans la clique.

La sélection du sommet candidat dans $C_0(K) \cup C_1(K)$ est faite avec une règle équivalente à la règle 4.4 avec le calcul du maximum étendu à tous les sommets de $C_1(K)$

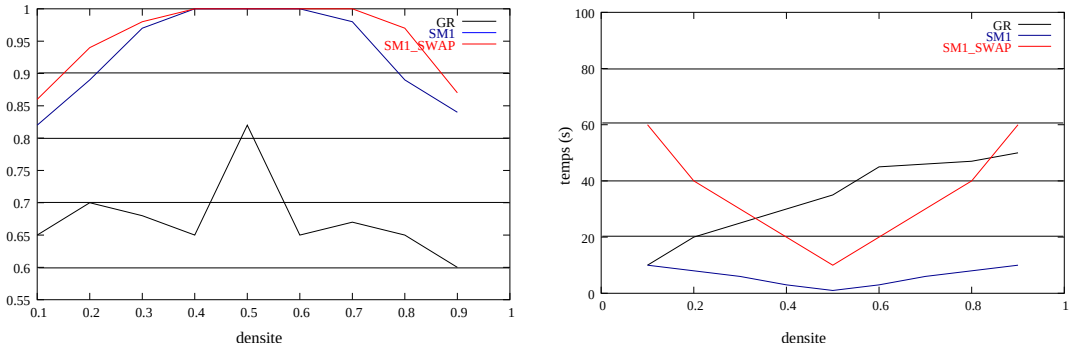
$$l := arg \max_{j \in C_0(K) \cup C_1(K)} |C_0(K) \cap N_j| \tag{4.6}$$

Le fait d'ajouter un sommet $j \in C_1(K)$ à la clique K nécessite alors de retirer de la clique l'unique sommet $k_l \in K$ tel que $(l, k_l) \notin E$.

La nouvelle clique obtenue est :

$$K = K \cup \{l\} - \{k_l\}$$

Le fait de choisir un sommet dans $C_1(K)$ permet de s'écarter de l'optimum local. Néanmoins il a été montré que ce choix n'améliore pas nécessairement l'optimum et ne rapproche pas toujours de la convergence. L'algorithme SM_SWAP peut également être amené dans certains cas à boucler. Pour éviter ce type de déconvenue, Fosser et al proposent de restreindre le choix du candidat potentiel à l'ensemble $C_0(K)$ durant un nombre fixe d'itérations mais également lorsque le nombre d'échanges est supérieur à un multiple de $|K|$ ou encore lorsque le sommet sélectionné est le même que celui supprimé par le dernier échange.



(a) $\frac{|K|}{\omega(G)}$ (b) temps d'exécution

FIG. 4.4 – Résultats expérimentaux sur les algorithme SM^1, SM_SWAP et un algorithme glouton classique .(a) représente le rapport entre la taille de la clique trouvée par rapport à la taille de la clique maximum en fonction de la densité du graphe de départ. (b) présente les temps de calcul nécessaires (en secondes) pour chacun des algorithmes, en fonction de la densité du graphe

Comme l'illustre la figure 4.4, l'algorithme SM_SWAP présente des résultats dont la qualité est proche de ceux proposés par SM^2 avec des temps d'exécution comparables à ceux de SM^1. En effet on peut voir sur la figure 4.4(a) que la taille de la clique obtenue par SM_SWAP et toujours très proche de la clique maximale et toujours supérieure à celle obtenue avec l'algorithme SM^1. Cet algorithme se présente donc comme une évolution majeure dans les algorithmes basés sur le calcul de la clique maximale.

4.2.3 Algorithme d'appariement de graphe bipartite et distance d'édition entre graphes

Bunke et all [RIESEN07] ont récemment proposé une approche, basée sur l'appariement de graphe bipartite, utilisant l'algorithme de Munkres, autrement appelé *algorithme hongrois*, qui permet de calculer efficacement la distance d'édition entre deux ensembles. L'idée de base est d'utiliser l'appariement d'ensembles de sommets pour définir une mesure de dissimilarité entre graphes.

L'algorithme proposé consiste, en premier lieu, à construire une matrice de coût $c_{i,j} = c(v_i \rightarrow u_j)$, prenant en compte uniquement les informations portées par les sommets (aucune information portant sur les arêtes adjacentes n'est utilisée à cette étape). Plutôt que d'utiliser l'algorithme de Munkres directement sur cette matrice, ce qui reviendrait à

ne pas prendre en compte les informations portées par les arêtes, une nouvelle matrice, tenant compte du voisinage, est calculée. Pour chaque couple de sommets u_i et v_j, un appariement minimum est calculé entre les arêtes des voisinages de ces deux sommets en utilisant de nouveau l'algorithme de Munkres. Pour chaque entrée de la matrice de coût initiale, l'algorithme de Munkres est ainsi utilisé pour apparier les arêtes adjacentes aux sommets u_i et v_j. On obtient la matrice de coût modifiée suivante :

$$C_{i,j} = c(v_i \rightarrow u_j) + \min\{\sum c(e_{v_i} \rightarrow e_{u_i})\} \tag{4.7}$$

avec $\min\{\sum c(e_{v_i} \rightarrow e_{u_i}\}$ calculé avec l'algorithme de Munkres.

L'algorithme de Munkres est alors appliqué sur la nouvelle matrice $C_{i,j}$ pour trouver un appariement global. Malgré le fait que l'appariement fourni par l'algorithme de Munkres est optimal en termes de sommets, le chemin d'édition fourni via cette méthode demeure sous-optimal. De fait, l'algorithme de Munkres fournit l'appariement de coût minimum, sans considérer les opérations d'édition possibles sur les arêtes. Ceci implique l'ajout de ces opérations à la fin du calcul pour retrouver la distance d'édition. Les sommets n'ayant pas été appariés devront également être supprimés, ainsi que les arêtes rattachées à ces sommets.

Les résultats expérimentaux tendent à prouver que cette méthode calcule un chemin d'édition sous-optimal de façon beaucoup plus rapide que les algorithmes exacts existants. La qualité des résultats par rapport aux algorithmes optimaux ne s'en trouve que peu affectée par rapport au gain obtenu en vitesse d'exécution.

4.3 Approches par optimisation

Le problème d'appariement de graphes peut également être résolu en utilisant des approches par optimisation. Ce type de méthode permet de redéfinir le problème en termes de minimisation/maximisation d'une fonction ou d'une énergie. Cette transformation du problème permet de bénéficier de tout l'arsenal des méthodes de recherche de la valeur minimimale/maximimale d'une fonction, telles que la programmation dynamique, la descente de gradient ou les approches par relaxation.

Si l'on considère un graphe $G = (V, E)$ avec $|V| = n$, ainsi qu'un sous-ensemble $C \subset V$. Nous pouvons définir le vecteur caractéristique de C par :

$$x^C = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \text{ avec } x_i = \begin{cases} \frac{1}{|C|} & \text{si } i \in C \\ 0 & \text{sinon} \end{cases} \tag{4.8}$$

Tout vecteur caractéristique appartient ainsi au simplex de dimension n :

$$\forall C \subset V \quad x^C \in S_n = \{x \in \mathbb{R}^n \,|\, e^t x = 1 \text{ et } \forall i \in \{1, \ldots, n\} \quad x_i \geq 0\} \tag{4.9}$$

Si l'on considère maintenant la matrice d'adjacence $A_G = (a_{i,j})$ d'un graphe G

$$a_{i,j} = \begin{cases} 1 & \text{si } (i,j) \in E, \\ 0 & \text{sinon.} \end{cases}$$

ainsi que la fonction

$$g(x) = x^t A_G x + \frac{1}{2} x^t x = x^t A x$$

$$\text{avec } \begin{cases} A = A_G + \frac{1}{2} I \\ x \in S_n \end{cases}$$

nous dirons que x^* est un maxima strict de g sur S_n si et seulement si :

$$\exists \epsilon > 0 \mid \begin{cases} \forall y \in S_n \quad | \quad |y - x^*| < \epsilon & : \quad f(y) \leq f(x^*) \\ \forall y \in S_n \quad | \quad |y - x^*| < \epsilon \text{ et } f(y) = f(x^*) & : \quad y = x^* \end{cases}$$

Théorème 4 Théorème de Motzkin-Straus(1965)

Soit $S \subset V$ et x^S son vecteur caractéristique alors :

1. *S est une clique maximum de G ssi x^S est un maximum global de g sur S_n. On a alors :*
$$\omega(G) = \frac{1}{2(1 - g(x^S))}$$
2. *S est une clique maximal de G ssi x^S est un maximum local de g sur S_n.*
3. *Tout maxima (local ou global) x de g sur S_n est strict et de la forme $x = x^S$ avec $S \subset V$.*

Intuitivement le théorème 4 dit qu'un sous-ensemble de sommet S d'un graphe G est une clique maximum si et seulement si son vecteur caractéristique x^S est un maxima de g sur le simplex S_n. En se basant sur le théorème 4, la recherche des maxima de $g(x) = x^t A x$ avec $A = A_G + \frac{1}{2} I$ peut être effectuée grâce à l'équation de replication suivante :

$$x_i(t+1) = x_i(t) \frac{(Ax(t))_i}{g(x)}$$

$$\sum_{i=1}^{n} x_i(t) = \frac{g(x)}{g(x)} = 1 \tag{4.10}$$

La solution converge ainsi vers un point stationnaire ($x_i(t+1) = x_i(t)$). Le théorème de Motzkin-Strausa servit de base à de nombreuses procédures de recherche de cliques [PARDAL90, BOMZE97, PELILL99] et a également servi pour démontrer les bornes théoriques sur la cardinalité de la clique maximum [WILF86, PARDAL90]. L'un des inconvénients majeurs associé avec la formulation originale de Motzkin-Straus tient dans le fait qu'elle peut en pratique fournir des solutions maximisant f, mais ne prenant pas la forme d'un vecteur caractéristique. Ce phénomène a été étudié empiriquement par Paradalos et Philips [PARDAL90] et formalisé par Pelillo et Jagoda [PELILL95].

Bomze propose en 1997 une solution à ce problème[BOMZE97] en considérant la fonction régularisée f suivante :

$$\begin{aligned} f(x) &= x^t \overline{A} x = x^t (A_{\overline{G}} + \tfrac{1}{2} I) x \\ &= \sum_{i=1}^{n} \tfrac{1}{2} x_i^2 + \sum_{j|(i,j)\notin E} x_i x_j \end{aligned}$$

Ce qui donnera pour un vecteur caractéristique x^C de $C \subset V$:

$$\begin{aligned} f(x^C) &= \tfrac{|C|}{2|C|^2} + \sum_{i=1}^{n} \sum_{j|(i,j)\notin E} x_i^C x_j^C \\ &= \tfrac{1}{2|C|} + \sum \sum_{(i,j)\in C^2 | (i,j)\notin E} x_i^C x_j^C \end{aligned}$$

Nous obtenons ainsi, si C est une clique, $f(x) = \frac{1}{2|C|}$. Plus la taille de la clique C est importante et plus la fonction $f(x)$ est «petite».

Ceci conduit à la reformulation suivante du théorème 4 :

Théorème 5 Théorème de Gibbons,Bomze(1997)

Soit $S \subset V$ et x^S son vecteur caractéristique alors :

1. *S est une clique maximum de G ssi x^S est un minimum global de f sur S_n. On a alors :*
$$\omega(G) = \frac{1}{2f(x^*)}$$
2. *S est une clique maximal de G ssi x^S est un minimum local de f sur S_n.*
3. *Tout minimum local (et donc global) x de g sur S_n est strict et de la forme $x = x^S$ avec $S \subset V$.*

L'autre intérêt de cette reformulation réside dans le fait qu'elle peut prendre en compte les graphes valués. En effet, la clique maximum correspond à la fonction de poids suivante :

$$\omega(G) = max\{|S| \text{ tel que } S \text{ est une clique de } G\}$$

alors que la clique maximum pondérée peut être définie grâce à un vecteur de poids $w \in \mathbb{R}^n$:

$$\omega(G, w) = max\{W(S) \text{ tel que } S \text{ est une clique de } G\}$$

avec :

$$W(S) = \sum_{i \in S} w_i$$

Avec cette reformulation les cliques de poids maximal (respectivement maximum) correspondent aux minimums locaux (respectivement global) de :

$$f(x) = x^t C(w) x$$

avec

$$C(w)_{i,j} = \begin{cases} \frac{1}{2w_i} & \text{si } i = j \\ \frac{1}{2w_i} + \frac{1}{2w_j} & \text{si } i \neq j \text{ et } (i,j) \notin E \\ 0 & \text{sinon} \end{cases}$$

Cette méthode permet ainsi d'introduire des informations a priori (distances entre points, similarités de régions, ...) pour restreindre la complexité du problème et guider l'algorithme dans ses choix. Comme dans le cas de graphes non pondérés, cet algorithme n'offre aucune garantie d'aboutir à un optimum global. Néanmoins les expériences portant sur le problème de la clique maximale, rapportées dans [BOMZE00], tendent à prouver que les bassins d'attraction des minimums globaux sont suffisamment larges pour espérer approcher un minimum global. Ces observations sont confirmées par les résultats des expériences menées par Pellilo [PELILL99].

4.3.1 Méthode de SoftAssign

Rangarajan et Gold proposent en 1996 une méthode d'appariement, basée sur une approche par optimisation, appelée *SoftAssign* [GOLD96, RANGAR97].

Nous allons pour décrire cette méthode, considérer 2 graphes, G et g, munis d'arêtes valuées.

Nous noterons alors $\begin{cases} G_{a,b} \text{ le poids de l'arête } (a,b) \text{ dans } G \\ g_{i,j} \text{ le poids de l'arête } (i,j) \text{ dans } g \end{cases}$

Soit M une matrice de permutation (Déf. 74) avec $M_{a,i} = 1$ si a est associé à i, 0 sinon. Il est tout d'abord nécessaire d'établir une fonction de distance C_{aibj} entre les attributs des arêtes des deux graphes

$$C_{a,b,i,j} = \begin{cases} 0 & \text{Si } G_{a,b} \text{ ou } g_{i,j} \text{ est null} \\ c(G_{a,b}, g_{i,j}) & \text{sinon} \end{cases}$$

qui pourra par exemple être de la forme suivante : $c(G_{a,b}, g_{i,j}) = 1 - 3|G_{a,b} - g_{i,j}|$

Dans ces conditions, Rangarajan va montrer que trouver un appariement revient à minimiser la fonction quadratique suivante :

$$Ewg(M) = -\frac{1}{2} \sum_{a=1}^{A} \sum_{i=1}^{I} \sum_{b=1}^{A} \sum_{j=1}^{I} M_{a,i} M_{b,j} C_{a,i,b,j} \tag{4.11}$$

avec A (respectivement I) représentant le nombre sommets dans G (respectivement g).

L'idée de base de cet algorithme est d'apparier un maximum d'arêtes qui se ressemblent. Lorsque qu'un sommet $a \in G$ est apparié avec un sommet $i \in g$, il peut être opportun d'apparier le sommet b (adjacent à a) avec le sommet j (adjacent à i). Ce raisonnement est basé sur la règle du rectangle (voir figure 4.5), qui consiste à obtenir un maximum de rectangles fermés $M_{a,i} M_{b,j} G_{a,b} g_{i,j}$.

Idée de la solution :

Le but est donc de minimiser $E_{wg}(M)$ par le biais d'un processus se déroulant en plusieurs étapes.

1. Choisir une valeur initiale de M

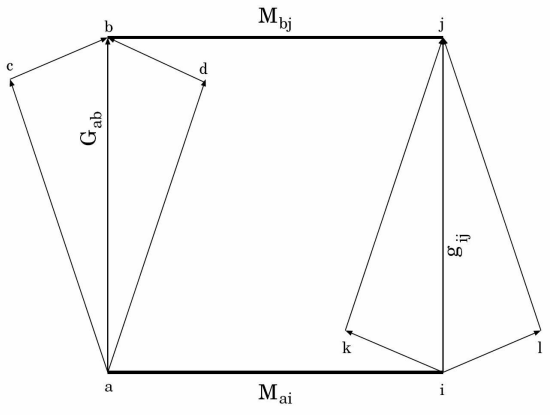


FIG. 4.5 – Règle du rectangle pour l'isomorphisme de sous-graphe

2. Faire une décomposition de Taylor de $E_{wg}(M)$.
   (a) Considérons $E_{wg}(M) = -\frac{1}{2}\sum_{a=1}^{A}\sum_{i=1}^{I}\sum_{b=1}^{A}\sum_{j=1}^{I} M_{a,i}M_{b,j}C_{a,i,b,j}$ comme une fonction de AI variables M_{ij}.
   (b) Appliquons Taylor à l'ordre 1 :

$$E_{wg}(M) \approx E_{wg}(M^0) - \sum_{a=1}^{A}\sum_{i=1}^{I}\frac{dE_{wg}(M)}{dM_{ai}}(M_{a,i} - M^0_{a,i})$$

   On posera alors $Q_{ai} = -\dfrac{dE_{wg}}{dM_{ai}} = \sum_{b=1}^{A}\sum_{j=1}^{I} M^0_{bj}C_{aibj}$

   Minimiser $E_{wg}(M)$ revient finalement à maximiser la somme $\sum_{a=1}^{A}\sum_{b=1}^{I} Q_{ai}M_{ai}$.

3. Faire un softassign correspondant au calcul du maximum de

$$\sum_{a=1}^{A}\sum_{i=1}^{I} Q_{a,i}M_{a,i}$$

4. prendre le M résultant et boucler en incrémentant le paramètre du Softassign β

En choisissant une bonne valeur initiale pour M, on pourra ainsi calculer le maximum de la somme $\sum_{a=1}^{A}\sum_{b=1}^{I} Q_{ai}M_{ai}$ avec l'algorithme SoftAssign, qui va converger vers la solution petit à petit (d'où son nom).

Même s'il a été démontré que les performances du SoftAssign diminuent drastiquement dès que les graphes à apparier sont trop différents (différence trop importante sur le nombre de nœuds dans chacun des deux graphes), il reste actuellement une des méthodes les plus efficaces. Une implémentation de cet algorithme est disponible à l'adresse : *http ://www.cise.ufl.edu/~anand/students/chui/research.html* avec de nombreux exemples d'utilisation de cette méthode.

4.3.2 Algorithme de SoftAssign à noyaux

Lozano et Escolano [LOZANO05] ont récemment amélioré les performances de l'algorithme de SoftAssign en proposant d'exploiter des résultats récents de la théorie spectrale sur graphes. Cette amélioration consiste à utiliser des noyaux de diffusion pour transformer un problème d'appariement de graphes non pondérés en un problème d'appariement de graphes pondérés.

L'idée de base part du principe que lorsque que l'on travaille avec des graphes non pondérés, le niveau d'ambiguïté structurel est tel qu'il est difficile de le lever uniquement en utilisant un algorithme de SoftAssign. Le fait d'utiliser des noyaux [KONDOR02, SMOLA03] sur les graphes permet d'obtenir des caractéristiques structurelles associées à chaque sommet, permettant de lever en partie les ambiguïtés. Les noyaux sur les graphes sont issus de résultats récents de la théorie de graphes [CHUNG97], en particulier les *noyaux de diffusion* fournissent une mesure de dissimilarité entre des paires de sommets appartenant à un même graphe. Dans le cas des noyaux de diffusion, une telle similarité peut être interprétée comme la probabilité qu'a un sommet d'être atteint à partir d'un autre, par le biais d'un processus de *marche aléatoire paresseuse* (*lazy randow walk*).

Un noyau K_G est donc donné par l'exponentiation matricielle de son Laplacien

$$K_G = e^{-\frac{\beta}{m}L_G}$$

avec $L_G = D_G - G$ représentant le Laplacien de G et D_G une matrice diagonale fournissant l'arité de chacun des nœuds. Le facteur de normalisation m, dépendant du nombre de sommets du graphe, a été introduit pour rendre les noyaux comparables, pour n'importe quel graphe. Pour un sommet i donné, la ligne $K_G(i,.)$ de la matrice K, correspond à la probabilité qu'a le sommet i d'atteindre chacun des autres sommets du graphe. $K_G(i,.)$ correspond donc à une densité de probabilité et on peut calculer son entropie $H^{K_G(i,.)}$.

La nouvelle définition de $C^K_{a,i,b,j}$ tient compte des noyaux des deux graphe K_G et K_g en tant qu'attribut structurel pour le sommet considéré. Le poids des arêtes est alors dépendant des entropies des probabilités de distribution qui sont associées aux deux sommets incidents, après le calcul d'un noyau.

$$C^K_{a,i,b,j} = G_{a,b}g_{ij}\delta_{a,i,b,j}K_{a,i,b,j} \tag{4.12}$$

avec

$$K_{a,i,b,j} = e^{-[(H_a^{K_G} - H_i^{K_g})^2 + (H_b^{K_G} - H_j^{K_g})^2]}$$

Le terme $\delta_{a,i,b,j}$ est égal à 1 si le sommet a est apparié avec l'arête b ainsi que le sommet i avec j, dans le cas contraire $\delta_{a,i,b,j}$ retourne -1. En pratique l'utilisation des noyaux dans la fonction de dissimilarité permet de pondérer, lors du processus d'appariement, les rectangles dont les sommets opposés ont des entropies comparables (voir figure. 4.5).

Des tests, effectués pour une application permettant de comparer la surface de différentes protéines, ont montré l'augmentation des performances de cet algorithme par rapport au SoftAssign classique. Il permet de maintenir une meilleure qualité des résultats, même lorsque les données ont subi de fortes altérations. Dans le cadre de l'utilisation avec les protéines, il a non seulement servi à les classer mais également à construire un prototype structurel propre à chacune des familles de protéines [LOZANO05].

4.4 Appariements entre partitions

Comme nous l'avons vu dans la section 4.1, la mise en correspondance d'images tient un rôle fondamental dans bon nombre d'applications. La correspondance entre les images est généralement effectuée par l'appariement de primitives de différentes dimensions :

- dimension 0 pour les points,
- dimension 1 pour les courbes ou
- dimension 2 pour les régions

Les processus de détection de primitives telles que les points ou les courbes sont dépendants de critères locaux, généralement peu robustes aux problèmes de bruit ou de données manquantes. Par contre les algorithmes de segmentation fournissent une partition de l'image en régions. Chaque région peut être associée à des caractéristiques complexes aussi bien géométriques que photométriques ou topologiques beaucoup plus robustes que des critères locaux.

Les algorithmes d'appariement basés régions associent en général à chaque région un vecteur de caractéristiques photométriques ou géométriques. De tels algorithmes ont tendance à ne pas prendre en compte les voisinages des régions alors que ceux ci représentent le contexte des deux régions à mettre en correspondance. Nous allons présenter dans les sections 4.4.3 et 4.4.2 des approches permettant de prendre en compte ces voisinages, en utilisant des algorithmes d'appariement de graphes planaires [NEUHAU04, LLADOS01] puis dans la section 4.4.4 une méthode permettant d'apparier des frontières en utilisant des systèmes de réécriture [GDALYA99A]. Nous présenterons ensuite (section 4.4.5) un algorithme basé également sur l'appariement de graphes planaires [CAIHUA95] et utilisant en plus les relations topologiques existant entre les régions, pour guider le processus de mise en correspondance. Enfin la section 4.4.6 présente une approche hiérarchique au problème de l'appariement de régions, proposées par Glantz [GLANTZ03]. Cette approche étudie le problème d'inconsistance existant entre les régions issues de deux segmentations, ne présentant en général pas le même degré de finesse.

4.4.1 Définitions et propriétés des graphes planaires

Il est nécessaire de rappeler quelques définitions et propriétés, relatives aux graphes planaires, indispensables à la poursuite de cette étude.

Définition 80 Sommet d'articulation *Soit $G = (V, E)$ un graphe. S'il existe un triplet de sommets distincts N_1,N_2 et $N_3 \in V^3$ tel que tout chemin connectant les sommets N_1 et N_2 passe forcement par le sommet N_3, alors N_3 est un appelé un* ***sommet d'articulation****.*

Un sommet d'articulation correspond dans le cas d'une partition à une région incluant une composante connexe. Ce type de sommet est illustré sur la figure 4.6 par le sommet D.

Définition 81 Graphe biconnecté

Un graphe G est dit ***biconnecté*** *si pour tout triplet de sommets distincts N_1,N_2 et N_3 de G, il existe un chemin entre N_1 et N_2 ne passant pas par le sommet N_3.*

Définition 82 Composante biconnectée

*Une **composante biconnectée** dans un graphe G est un sous-graphe maximal de G qui est biconnecté.*

Définition 83 Sommet de biarticulation

*Soit $G = (V, E)$ un graphe. S'il existe un quadruplet de sommets distincts N_1,N_2,N_3,$N_4 \in V^4$ tel que tout chemin connectant les sommets N_1 et N_2 passe forcement par l'un des sommets N_3 ou N_4, alors N_3 et N_4 sont appelés une paire de **sommets de biarticulation**.*

Les sommets A et B présentés sur la figure 4.6 sont des sommets de biarticulation.

Définition 84 Graphe triconnecté

*Un graphe G est dit **triconnecté** si pour tout quadruplet de sommets distincts N_1,N_2,N_3,N_4 de G, il existe un chemin entre N_1 et N_2 ne passant pas par les sommets N_3 ou N_4.*

Définition 85 Composante triconnectée

*Une **composante triconnectée** dans un graphe G est un sous-graphe maximal de G qui est triconnecté.*

Une composante biconnectée peut être partitionnée en composantes triconnectées elles-mêmes biconnectées par des sommets de biarticulation.

Un graphe planaire possède la propriété de pouvoir être dessiné sur un plan, sans qu'aucune de ses arêtes ne se croise. Un tel dessin représente un **plongement** de ce graphe dans le plan.

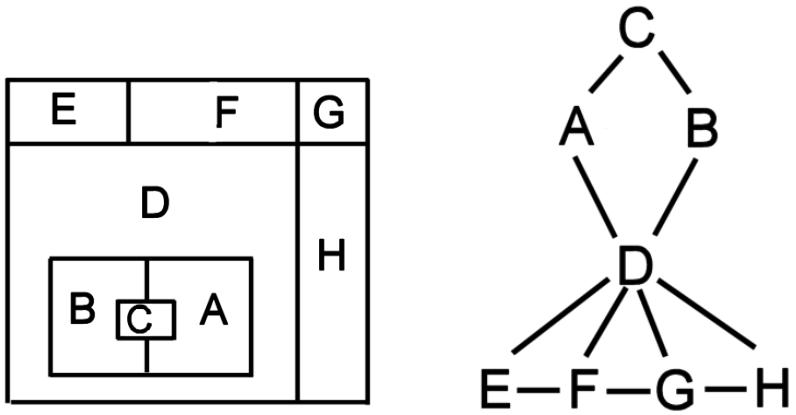
Propriété 2 *[*HOPCRO72*]*

Un graphe planaire triplement connecté admet exactement deux plongements différents dans un plan. L'un des deux plongements est obtenu à partir de l'autre en inversant simplement l'ordre de toutes les arêtes autour de chaque sommet.

*On dit alors que ce plongement correspond au **miroir** de l'autre.*

Dans le cadre de l'appariement de régions, si l'on exclut le cas de figure dans lequel une des images à apparier est le miroir de l'autre, chaque composante planaire triconnectée (Déf. 85) admet un unique plongement dans le plan (voir Prop. 2). Ceci implique que si un sommet d'une composante triconnectée est apparié avec un sommet d'une composante triconnectée dans l'autre graphe, alors leurs arêtes respectives ne pourront être appariées que dans un certain ordre.

Par exemple si l'on considère un sommet $N \in G$, appartenant à une composante triconnectée, apparié avec un sommet $N' \in G'$, appartenant également à une composante triconnectée . Si l'on désigne par $(N_1, \ldots, N_p)$ (resp. $(N'_1, \ldots, N'_q)$ l'ensemble des voisins de N (resp. N') ordonnés dans le sens des aiguilles d'une montre, le plongement des deux graphes induit la règle suivante :



(a) partition (b) RAG

FIG. 4.6 – Une partition avec son RAG correspondant. Le sommet D est un sommet d'articulation alors que les sommets A et B sont des sommets de biarticulation.

$$N_i \leftrightarrow N'_j \quad : \quad N_{i+k} \leftrightarrow N'_{j+k} \tag{4.13}$$

avec $\leftrightarrow$ représentant l'appariement des sommets N_i et N'_j et $k > 0$. Cette règle est appelée la *préservation de l'ordre*.

4.4.2 Algorithme d'appariement inexact de graphe planaire

Bunke et Neuhaus [NEUHAU04] proposent un algorithme permettant de mettre en correspondance deux graphes planaires, plongés dans le plan, en calculant un chemin d'édition minimal entre ces deux graphes.

Pour calculer une séquence d'édition minimale, ils définissent le voisinage $N(u)$ d'un sommet u comme le sous-graphe constitué du sommet u, de tous les sommets connectés à u ainsi que de toutes les arêtes reliant ces sommets. Plus formellement, soit $G = (V, E, \alpha, \beta)$ un graphe avec V l'ensemble de ses sommets, E l'ensemble de ses arêtes et α et β deux fonctions de label pour les sommets et les arêtes. Le voisinage $N(u)$ d'un sommet u dans G est défini comme le sous graphe induit $N(u) = (V_u, E_u, \alpha_u, \beta_u)$ avec

$$\begin{aligned} V_u &= \{u\} \cup \{v \in V | (u,v) \in E \text{ ou } (v,u) \in E\} \\ E_u &= E \cap (V_u \times V_u) \\ \alpha_u &= \alpha|_{V_U} \\ \beta_u &= \beta|_{E_U} \end{aligned} \tag{4.14}$$

A partir de deux graphes planaires $G = (V, E, \alpha, \beta)$ et $G' = (V', E', \alpha', \beta')$, l'algorithme proposé se déroule avec les sept étapes suivantes :

(i) Sélectionner deux sommets initiaux $u_0 \in V$ et $u'_0 \in V'$.

(ii) Ajouter à une pile FIFO Q la substitution $u_0 \rightarrow u'_0$

(iii) Dépiler de Q la prochaine substitution $u \rightarrow u'$

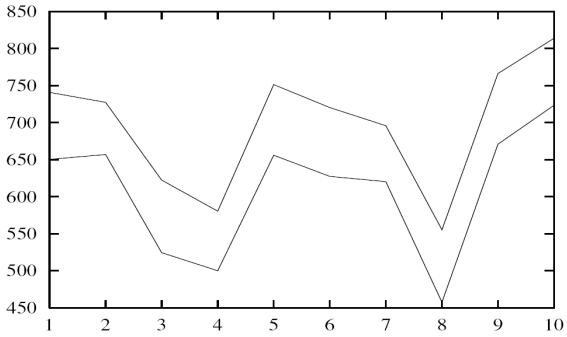


FIG. 4.7 – Distance d'édition exacte (en bas) ainsi que la distance d'édition approximée (courbe du haut) pour 10 graphes et sous-graphes avec 10 sommets.

(iv) Effectuer l'appariement des voisinages $N(u)$ et $N(u')$

(v) Ajouter les nouvelles substitutions apparaissant à l'étape (iv) dans Q

(vi) Si Q n'est pas vide, aller à l'étape (iii)

(vii) Effacer de G et G' tous les sommets et arêtes n'ayant pas été traités

Cet algorithme fournit un appariement entre les deux graphes G et G' avec la distance d'édition correspondante $d(G, G')$. L'étape (iv) de cet algorithme consiste à apparier les voisinages de deux sommets. Le fait d'utiliser des graphes plongés permet de considérer l'ensemble des sommets adjacents au sommet central comme une séquence ordonnée. Au lieu de regarder ces voisinages comme des graphes, ils peuvent alors être considérés comme des séquences de sommets ordonnées pouvant être alignés par des méthodes d'appariement de chaînes circulaires [BUNKE93B, LLADOS01, PERIS02, MOLLIN02, ROBLES02] (voir Section 4.4.1). En considérant des graphes dont l'arité des sommets est bornée par M, ce type de procédure admet une complexité en $\mathcal{O}(M^2)$.

Lorsque l'algorithme se termine, il fournit une séquence d'édition valide permettant de borner la véritable distance d'édition entre les deux graphes de départ. La séquence d'édition fournie étant fortement dépendante des deux sommets sélectionnés lors de l'étape (i), il est nécessaire d'effectuer plusieurs fois l'algorithme en sélectionnant des sommets de départ différents, pour améliorer le coût de la séquence d'édition renvoyée. Le choix de ce couple de départ peut par exemple être effectué en prenant les sommets les plus proches des barycentres correspondant aux plongements des deux graphes à apparier. De façon plus générale, toute connaissance a priori sur l'application peut servir pour guider le choix du couple de départ.

Comme l'illustre la figure 4.7, les résultats expérimentaux montrent que la distance approximée par cet algorithme est fortement corrélée à la distance d'édition exacte et ces deux distances semblent même correspondre à un facteur additif constant près. Ces observations sont validées [NEUHAU04] par des tests effectués sur la base NIST-4, donnant un facteur de corrélation, entre les deux méthodes, de $r = 0.99$ pour des graphes et sous-graphes comptant 10 sommets et de $r = 0.85$ pour des graphes avec 100 sommets.

L'utilisation d'une file ou d'un arbre de recherche ainsi que l'utilisation de l'orientation lors de l'appariement de sommets sont des heuristiques systématiquement utilisées dans le cadre de l'appariement de partitions. Nous les reverrons donc sous différentes formes dans les section suivantes.

4.4.3 Appariement de sous-graphes basé sur les frontières des régions

Llados [LLADÓS97, LLADOS01] propose un algorithme d'appariement de sous-graphes permettant de prendre en compte l'orientation du plan d'une part, mais également les voisinages des deux régions à apparier.

En considérant les deux graphes d'adjacence de région (RAG) , issus des deux partitions à apparier : un graphe modèle $G_M = (V_M, E_M)$ ainsi qu'un graphe d'entrée $G_I = (V_I, E_I)$, Llados propose un algorithme permettant de calculer la séquence d'édition minimum permettant de transformer un RAG en l'autre, de façon à les rendre isomorphes.

L'idée de base de cet algorithme consiste non pas à mettre en correspondance directement des régions, mais à apparier les frontières de ces régions en utilisant un algorithme d'appariement de chaînes circulaires puis à faire grandir ces chaînes en effectuant des fusions de régions.

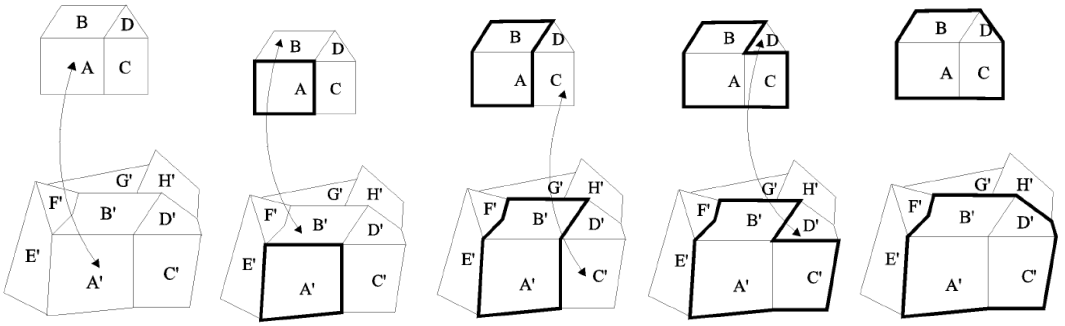
La première étape de cet algorithme consiste à mettre en correspondance une région initiale du graphe modèle G_M avec une région du graphe d'entrée G_I . Leurs frontières sont alors représentées par des chaînes circulaires s_M et s_I, qui sont appariées en utilisant un algorithme d'appariement de chaînes circulaires [MAES90, BUNKE93A, LLADOS01, PERIS02, MOLLIN02, ROBLES02]. Cet algorithme permet de calculer une distance d'édition circulaire entre les frontières des deux régions sélectionnées.

Llados définit des coûts pour les différentes opérations d'édition. Le coût de **substitution** entre un nœud $r_M \in V_M$ et un nœud $r_I \in V_I$, noté $r_M \rightarrow r_I$, sera donné par le coût d'appariement de leurs frontières respectives $\delta(r_M)$ et $\delta(r_I)$:

$$c(r_M \rightarrow r_I) = d_c(\delta(r_M), \delta(r_I)))$$

avec d_c correspondant à la distance d'édition circulaire entre deux chaînes [MAES91].

Ce processus d'appariement des frontières des régions utilise une distance d'édition circulaire entre chaînes comparable à l'étape iv de l'algorithme de Bunke et al [NEUHAU04] (section 4.4.2).



(a) $\{\}$ (b) $\{A \rightarrow A'\}$ (c) $\{A \rightarrow A', B \rightarrow B'\}$ (d) $\{A \rightarrow A', B \rightarrow B', C \rightarrow C'\}$ (e) $\{A \rightarrow A', B \rightarrow B', C \rightarrow C', D \rightarrow D'\}$

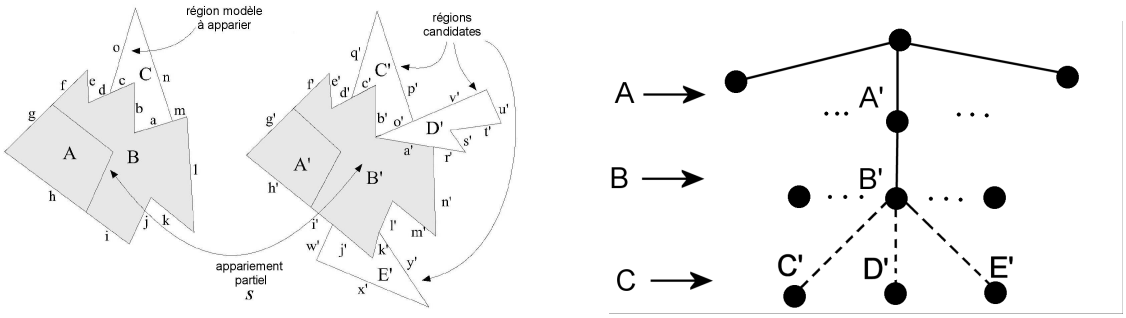
FIG. 4.8 – Exemple de croissance de chaînes en terme de similarité de voisinage.

L'une des particularités de l'approche de Llados tient dans la définition d'un coût de **décalage**. Ce décalage s'effectue pour un nœud $r_I^j \in V_I$ adjacent à un nœud $r_I^l \in V_I$, le long de la frontière de r_I^l. Ce décalage est noté $r_I^l or_I^j[X,Y]$, avec X correspondant à la sous chaîne partagée par r_I^l et r_I^j et Y celle qui sera partagée par r_I^l et r_I^j une fois le décalage de r_I^j effectué. Le coût affecté à ce décalage est calculé par rapport au pourcentage de recouvrement entre X et Y :

$$c(r_I^l or_I^j[X,Y]) = 1 - \frac{l_{XY}}{\max(l_X, l_Y)} \tag{4.15}$$

avec l_X et l_Y représentant les longueurs respectives de X et Y et l_{XY} celle de la sous-chaîne commune à X et à Y. L'opération de décalage est utilisée principalement pour permettre la préservation des relations structurelles entre les régions en dépit de petites perturbations.

La figure 4.9(a) illustre une opération pour l'appariement partiel ayant mis en correspondance les régions AB et $A'B'$. A cette étape il y a trois successeurs possibles pouvant s'apparier à la région C ($C \rightarrow C'$,$C \rightarrow D'$ ou bien $C \rightarrow E'$). Ces trois possibilités auraient un coup identique si seule l'opération de substitution était envisagée. L'opération de décalage ajoute des informations structurelles permettant de résoudre ce conflit. Si l'on analyse par exemple l'appariement $C \rightarrow C'$, C est adjacente à AB par la sous chaîne abc et C' est adjacente à $A'B'$ par la sous chaîne $b'c'$. Pour améliorer la distance d'édition, le mieux serait d'apparier la chaîne abc avec $a'b'c'$, mais il faudrait pour cela que la région C' soit adjacente à $A'B'$ sur la frontière $a'b'c'$. Pour transformer le graphe modèle un décalage doit être effectué sur la région C lors de sa substitution par la région C'. En utilisant la notation définie équation 4.15 ce décalage est noté $A'B'oC'[b'c', a'b'c']$. Le coût $c(A'B'oC'[b'c', a'b'c'])$ est défini par l'équation 4.15 en prenant en compte le pourcentage de recouvrement entre les deux chaînes. Les coûts $c(A'B'oD'[a', a'b'c'])$ et $c(A'B'oE'[j'k', a'b'c'])$ sont calculés de manière analogue et les trois états intermédiaires possibles sont générés.



(a) opération de décalage (b) arbre d'états

FIG. 4.9 – (a) Illustration de l'opération de décalage. (b) État intermédiaire correspondant à l'appariement de AB avec $A'B'$. Les 3 extensions possibles sont $C \rightarrow C'$,$C \rightarrow D'$ et $C \rightarrow E'$.

En considérant ces différentes opérations, lorsqu'un appariement courant est élargi par la mise en correspondance d'une région r_M avec r_I, le coût de cette extension est calculé en utilisant l'expression suivante :

$$\begin{aligned} c(r_M \rightarrow r_I) =& W_1 c(r_M \rightarrow r_I) \\ &+ W_2 \max\left(c(R_I o r_I[X,Y]), c(r_I o R_I[X,Z])\right) \\ &+ W_3 \left| \frac{R_M}{R_I} - \frac{r_M}{r_I} \right| \end{aligned} \tag{4.16}$$

avec W_1, W_2, W_3 des facteurs de pondération, X la sous chaîne partagée par R_I et r_i, Y celle partagé par R_I et r_i après avoir décalé r_I autour de R_I et Z la sous chaîne partagée par R_I et r_i après avoir décalé R_I autour de r_I. Les deux termes $\frac{R_M}{R_I}$ ainsi que $\frac{r_M}{r_I}$ représentant les rapports entre les aires des régions appariées, tendent à favoriser l'appariement de régions ayant des surfaces de tailles comparables.

Cet algorithme permet de trouver un morphisme de sous graphe, en reformulant le problème en terme de distance d'édition sur les frontières des régions. La prise en compte du voisinage lors du processus d'appariement, en mettant en correspondance les frontières des régions, permet de guider l'algorithme alors que l'utilisation de l'orientation du plan permet de réduire de façon conséquente la complexité du problème. En effet, alors que ce problème est théoriquement NP-complet, expérimentalement cet algorithme fournit une solution en un temps polynomial. Les opérations d'édition autorisées permettent également d'apparier des graphes ayant subi des distorsions dues au bruit présent dans les images sans toutefois permettre de résoudre de façon efficace l'inconsistance inhérente à tout processus de segmentation.

4.4.4 Appariement hiérarchique de courbes utilisant des réécritures

Gdalyahu et al [GDALYA99A] proposent en 1999 un algorithme permettant de mettre en correspondance des courbes représentant des contours d'objets segmentés. La particularité de cette approche tient au fait qu'elle permet la mise en correspondance de frontières représentées à différents niveaux de finesse par le biais de systèmes de réécriture. Les courbes représentant les frontières à mettre en correspondance, sont représentées par une suite de segments.

Gdalyahu utilise un algorithme d'appariement de chaînes circulaires [MAES90, BUNKE93A, LLADOS01, PERIS02, MOLLIN02, ROBLES02] prenant en compte l'orientation et la longueur des segments pour calculer la distance d'édition entre les courbes frontières. Pour rendre son algorithme plus robuste au bruit et aux perturbations locales, il propose une opération consistant à fusionner plusieurs segments, permettant ainsi de simplifier la représentation du contour de façon locale et adaptative en passant d'une échelle fine à une échelle plus grossière (voir figure 4.10). Deux segments adjacents peuvent ainsi être remplacés par un seul segment connectant les deux points terminaux les plus éloignés.

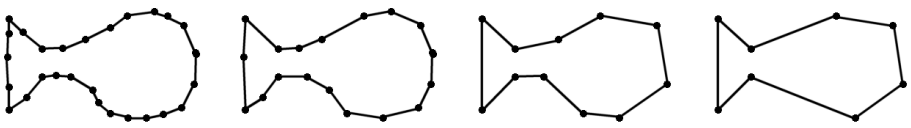


FIG. 4.10 – Contour d'un objet représenté à différents niveaux de détails.

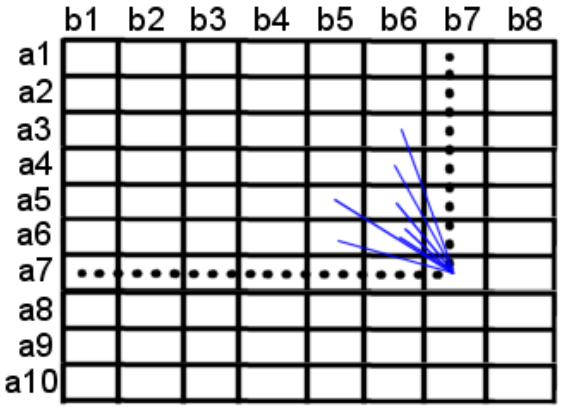


FIG. 4.11 – Chaque entrée $R[k,l]$ de la matrice de coût est calculée avec l'équation 4.17. Les lignes (bleues) repésentent les opérations de fusion (incluant la substitution classique) alors que les lignes pointillées matérialisent les suppressions

Considérons les deux séquences ordonnées $A = \{a_1, \ldots, a_N\}$ et $A' = \{a'_{1'}, \ldots, a'_{N'}\}$. Classiquement, les coûts correpondant aux appariements partiels sont stockés dans une matrice de coût R dans laquelle une entrée $R[\mu,\nu]$ correspond au coût de l'appariement de μ segments de A avec ν segments de A'.

Supposons que la matrice de coût R est déjà partiellement remplie entre $R[1,1]$ et $R[\mu,\nu]$. A chaque étape l'algorithme va étendre R en remplissant une nouvelle ligne (de longueur ν) et/ou une nouvelle colonne (de longueur μ). Une colonne (resp. ligne) sera ainsi ajoutée si la valeur minimum de R est dans la dernière colonne (resp. ligne) calculée, alors que les deux seront ajoutées si le minimum est obtenu dans la case correspondant au dernier coin calculé.

Contrairement aux algorithmes d'édition classiques, Gdalyahu introduit dans son calcul de coût un terme calculé sur un domaine Ω, généralisant l'opération de substitution à la substitution de segments fusionnés. Le domaine Ω correspond à une partie triangulaire de la matrice R définie par :

$$\Omega = \{\alpha, \beta \quad | \quad 0 < \alpha < k, 0 < \beta < l, (k-\alpha) + (l-\beta) \leq K\}$$

Le calcul de ce terme implique $K(K-1)/2$ évaluations pour considérer l'ensemble des fusions alternatives possibles. Les domaines de minimisation pour les opérations correspondant à la suppression de plusieurs éléments, en ligne ou en colonne sont associés à un coût d'interruption (w_3) ainsi qu'à une récompense (w_2), permettant de mettre en compétition ces opérations avec celle de substitution. Notons que l'opération de suppression d'un seul élément, utilisée dans les algorithmes d'appariement circulaire classiques, est incluse dans ce modèle et correspond au cas $K = 2$.

Chaque nouvelle entrée $R[k,l]$, lors de l'extension d'un bloc de taille $\mu \times \nu$, sera calculée en utilisant les expressions suivantes.

$$R[k,l] = \min\{r_1, r_2, r_3\} \tag{4.17}$$

avec

$$r_1 = \min_{\alpha,\beta\in\Omega}\{R[\alpha,\beta] - S(\alpha k, \beta l)\} \quad (4.18)$$

$$r_2 = \min_{0<\alpha<k}\{R[\alpha,l] + w_3 - w_2.(k-\alpha)\} \quad (4.19)$$

$$r_3 = \min_{0<\beta<l}\{R[k,\beta] + w_3 - w_2.(l-\beta)\} \quad (4.20)$$

Le terme r1 correspond à une substitution de $(k-\alpha)$ élements de la première chaîne avec $(l-\beta)$ éléments de la seconde alors que r2 (resp. r3) correspond à la suppression de $(k-\alpha)$ (resp. $(l-\beta)$) élements de la première (resp. deuxième) chaîne.

L'approche proposée par Gdalyahu permet donc de mettre en correspondance deux frontières représentées par une suite de segments. La particularité de cette approche est qu'elle permet de gérer les perturbations locales en incluant des opérations de concaténations (réécritures) offrant une représentation multi-échelle des chaînes à apparier. Des coûts spécifiques sont attribués à ces nouvelles opérations qui sont désormais prises en compte dans le calcul de la distance d'édition. L'inconvénient majeur de cette méthode tient dans le fait qu'elle nécessite d'envisager toutes les réécritures possibles pour chaque frontière. En effet les opérations consistant à fusionner des segments de frontière ne sont aucunement guidées par les données du fait qu'elle ne tiennent absolument pas compte du voisinage de la région considérée. Elles doivent toutes être prises en compte, ce qui implique énormément de temps de calcul dès que les longueurs des chaînes à apparier deviennent importantes. Cet algorithme n'est donc pas adapté à une utilisation avec des partitions.

4.4.5 Appariement inexact de graphes sur les relations topologiques

Pour effectuer la mise en correspondance de régions, Wang propose de prendre en compte l'aspect planaire de la partition en effectuant l'appariement inexact de graphes planaires [CAIHUA95]. L'originalité de ce travail tient à la prise en compte des relations topologiques existant entre les régions et notamment les relations d'inclusion.

Wang propose d'effectuer une séquence d'opérations d'édition sur un des deux graphes, de façon à le rendre isomorphe à l'autre, en effectuant un traitement spécifique pour les sommets d'articulation (Déf. 80) codant les relations d'inclusion.

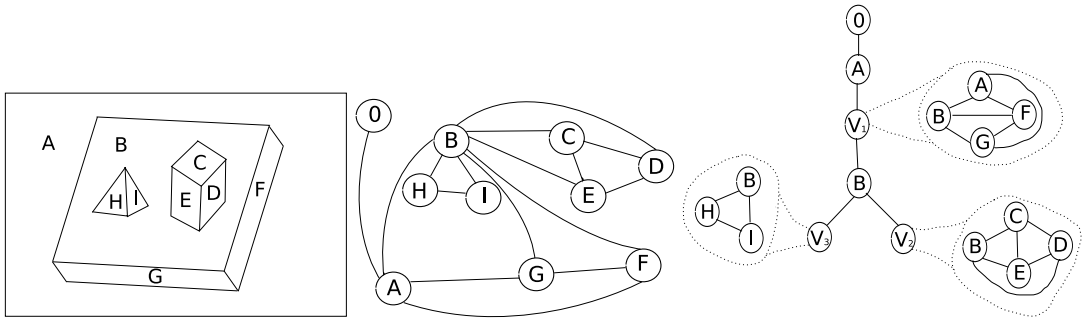
Pour guider le processus d'appariement, Wang propose de partitionner le RAG en composantes biconnectées (Déf 82) représentées dans un arbre. Il définit ainsi une structure appelée BRAG :

Définition 86 BRAG

*Soient N_i un sommet appartenant à une composante biconnectée B dans un graphe G et $S(N_i) = \{N_{i_1}, N_{i_2}, \ldots, N_{i_m}\}$ l'ensemble des sommets rencontrés en tournant dans le sens des aiguilles d'une montre dans B autour de N_i. Le sous-graphe composé des sommets $\{N_i\} \cup S(N_i)$ et des arêtes reliant ces sommets est appelé un **BRAG** (ou Basic RAG) du sommet N_i.*

Avec ce type de partitionnement, les sommets d'articulation (Déf. 80) apparaissent non seulement comme sommets dans l'arbre mais également dans les deux composantes

biconnectées incidentes au sommet d'articulation. La figure 4.12 illustre ce type de partitionnement en arbre en proposant le RAG correspondant à la partition de la figure 4.12(a) ainsi que l'arbre sur la figure 4.12(c).



(a) régions dans une image (b) RAG correspondant (c) partitionnement du RAG en arbre

FIG. 4.12 – Un graphe planaire (b) décrivant les régions et leurs relations (0 correspond à l'extérieur de l'image avec l'arbre correspondant (c)

L'algorithme, proposé par Wang, commence par sélectionner une paire de sommets (N_i, N'_j), appartenant à chacun des deux graphes, dont les caractéristiques sont proches et qui ne sont pas des sommets d'articulation. A partir de ce point de départ, l'algorithme transforme les deux RAG en deux graphes isomorphes en effectuant les opérations d'édition impliquant les coûts minimaux. Les appariements optimaux sont construits en examinant toutes les paires de sommets de départ envisageables.

En partant de la paire de sommets (N_i, N'_j), les deux graphes, contenant ces sommets, sont transformés en deux graphes isomorphes. Ceci est effectué en construisant itérativement les BRAG isomorphes, pour les sommets appariés via les étapes suivantes :

1. Construire les deux BRAG correspondant aux sommets N_i et N'_j et
2. les apparier en effectuant des opérations d'édition de coût minimal.
3. Les paires de sommets $\{N_{i_k}, N'_{j_k} \quad | \quad k = 1, \ldots, n\}$ appariées par l'étape 1, sont ajoutées à une file Q.
4. Extraire de la file Q une paire de sommets (N_{i_k}, N'_{j_k}) n'ayant pas encore été traitée. Si N_{i_k} ou N'_{j_k} est un sommet d'articulation, **aller à l'étape 7**.
5. Construire deux BRAG isomorphes pour les sommets N_{i_k} et N'_{j_k}, indépendamment de l'appariement obtenu précédemment. Si les paires de sommets appariés lors de cette étape n'entrent pas en conflit avec les mises en correspondances précédentes, l'appariement courant est étendu par l'ajout des deux BRAG de N_{i_k} et N'_{j_k}. Les nouveaux appariements obtenus sont ajoutés à la file Q·
6. Si un appariement venant d'être obtenu présente un conflit avec un autre obtenu précédemment, un processus pour gérer ce conflit est appliqué pour ajuster le nouvel appariement, en supprimant des arêtes gênantes.
7. La paire (N_{i_k}, N'_{j_k}) est marquée comme ayant été traitée.
8. Itérer les étapes 4 à 7 jusqu'à ce que toutes les paires de sommets dans Q aient été traitées.

Cet algorithme prend en compte l'orientation du plan et fournit les sommets dans l'ordre horaire où ils apparaissent, le coût d'appariement minimal pour deux BRAG peut être à nouveau calculé à l'aide d'un algorithme d'appariement de chaînes circulaires [MAES90, BUNKE93A, LLADOS01, PERIS02, MOLLIN02, ROBLES02].

Cette approche est très similaire à l'approche classique proposée par Bunke [NEUHAU04] (section 4.4.2) mais permet en plus de prendre en compte les relations topologiques existant entre les régions. Les étapes 3 et 4 de cet algorithme sont équivalentes aux étapes ii et iii de l'algorithme de Bunke. La principale évolution de cette méthode tient dans le traitement spécifique pour les sommets d'articulation. En effet alors que les autres types de sommets continuent à être appariés en suivant l'ordre circulaire relatif à leur positionnement autour d'un sommet principal (voir les sommets E,F,G et H de la figure 4.6), les sommets d'articulation comme le sommet D sur la figure 4.6 n'induisent pas de contrainte sur la préservation de l'ordre et sont traités spécifiquement.

Supposons que la construction de deux graphes planaires commence avec la mise en correspondance des deux sommets N_{i_1} et N'_{j_1}. Dans ce cas, $C(k,l)$ qui représente l'appariement de coût minimal permettant de rendre les deux sous-graphes $\{N_i, N_{i_1}, N_{i_2}, \ldots, N_{i_k}\}$ et $\{N_{j_1}, N_{j_2}, \ldots, N_{j_l}\}$ isomorphes est alors calculé récursivement de la façon suivante :

$$\begin{aligned} C(k,l) = min\{&C(k-1,k-1) + Cmt(N_{i_k}, N'_{j_l}),\\ &C(k,l-1) + Cde(N'_j, N'_{j_l}),\\ &C(k-1,l) + Cde(N_i, N_{i_k)},\\ &C(k-1,l) + Cdn(N_{i_k}),\\ &C(k,l-1) + Cdn(N'_{j_l}),\\ &C(k-1,l-1) + Cdm(N_{i_k}),\\ &C(k-1,l-1) + Cdm(N'_{j_l}),\\ &C(k-1,l) + Cmg(N_{i_k}),\\ &C(k,l-1) + Cmg(N'_{j_l})\} \end{aligned} \tag{4.21}$$

avec C(0,0) étant initialisé à 0. $Cmt(N_{i_k}, N'_{j_l})$ représente le coût de la substitution (ou de l'appariement) de N_{i_k} avec N'_{j_l}, Cde et Cdn les coûts de suppression des arêtes et des sommets alors que Cmg représente le coût de fusion de sommets. Cdm représente le coût de suppression d'un sommet subissant une occlusion due à un sommet adjacent.

La méthode proposée par Wang permet d'obtenir un appariement inexact de deux graphes planaires valués. Le principal avantage de cette méthode est qu'en plus de tirer parti de l'orientation du plan en mettant à profit les propriétés relatives aux graphes planaires, elle utilise les relations topologiques entre les régions pour guider le processus d'appariement. La caractérisation des nœuds d'articulation, et le partitionnement du RAG qui en découle permet de guider le processus d'appariement en fonction des relations d'inclusion entre les régions. L'inconvénient majeur de cet algorithme réside dans le fait qu'il nécessite d'itérer le processus d'appariement sur un nombre important de couples de départ, ce qui peut amener à des temps de calcul relativement élevés pour aboutir à la solution optimale.

4.4.6 Appariement hiérarchique de régions

Le bruit ou les distorsions pouvant affecter les images conduisent parfois à de mauvaises segmentations des objets présents dans celles-ci. Il est en effet très rare d'obtenir une correspondance exacte entre une région et un véritable objet. Un objet est, le plus souvent, défini par l'union de plusieurs régions et inversement une région peut très bien englober plusieurs objets. Les méthodes présentées dans les section 4.4.3 à 4.4.5 ne permettent pas de surpasser ce problème facilement, les régions à apparier n'étant représentées qu'à un seul niveau de détail. La figure 4.13 illustre ce problème d'inconsistance en présentant deux segmentations d'une même zone dans lesquelles une région se trouve être sur-segmentée d'un coté par rapport à l'autre.

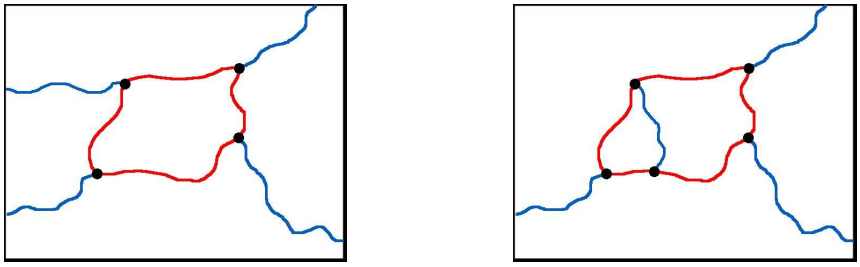


FIG. 4.13 – Exemple du problème d'inconsistance pour la segmentation. La région centrale de l'image de gauche se trouve être sur-segmentée sur l'image de droite.

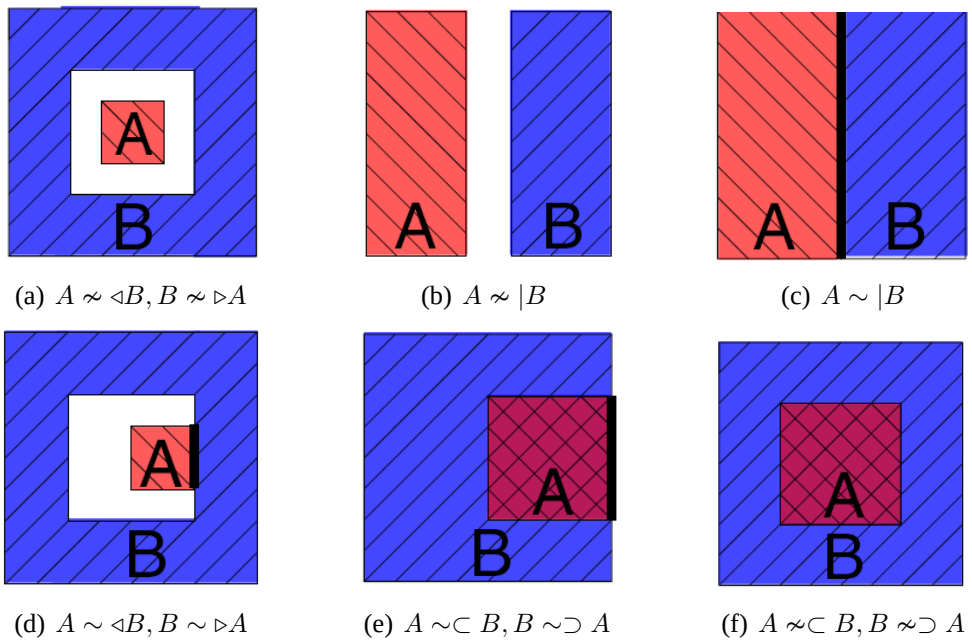
Pour pallier à ce problème d'inconsistance entre les régions présentes dans les deux segmentations, Glantz et Pelillo proposent de considérer une hiérarchie complète de partitions [GLANTZ03] plutôt qu'une simple segmentation correspondant à un niveau particulier de finesse. Dans le cadre de la reconnaissance, l'idée d'utiliser des hiérarchies d'images est renforcée par le fait que les objets admettent généralement une représentation hiérarchisée assez naturelle.

Cette approche propose de construire un graphe d'association (voir Déf. 76) à partir de deux hiérarchies de partitions, dans lesquelles les sommets indiquent les appariements potentiels entre deux régions alors que les arêtes représentent la consistance d'un point de vue topologique. Avec cette approche, un appariement consistant avec un score de similarité maximum entre les régions appariées correspond à une clique de poids maximum dans le graphe d'association (voir section 4.1.1).

L'étape qui consiste à construire les deux hiérarchies de segmentations $(G_i, \overline{V_i}, g_i)_{i=0}^{m}$ avec $G_i = (V_i, E_i, \iota_i)$ pour tout niveau i est effectuée par le biais d'un processus de segmentation hiérarchique par contractions successives de graphes duaux [WILLER94, HAXHIM03] (voir sections 2.3.2 et 2.4.3.b).

Le graphe d'association construit à partir des deux hiérarchies prend en compte différentes relations topologiques existant entre les régions. Glantz et al combinent différentes relations telles que la relation de voisinage, la relation d'inclusion ou encore le fait qu'une région entoure une autre. Pour deux régions distinctes $r_i \neq r_j$, on peut définir cinq relations différentes liant ces deux régions, toutes ces relations s'excluant l'une l'autre :

1. $r_i \subset r_j$ lorsque r_i est inclue dans r_j



(a) $A \nsim \triangleleft B, B \nsim \triangleright A$ (b) $A \nsim |B$ (c) $A \sim |B$

(d) $A \sim \triangleleft B, B \sim \triangleright A$ (e) $A \sim \subset B, B \sim \supset A$ (f) $A \nsim \subset B, B \nsim \supset A$

FIG. 4.14 – Les 10 relations topologiques différentes. Les lignes épaisses représentent les arêtes définissant une relation de voisinage.

2. $r_i \supset r_j$ lorsque r_i inclut r_j
3. $r_i \triangleright r_j$ lorsque r_i est entourée par r_j
4. $r_i \triangleleft r_j$ lorsque r_i entoure r_j
5. $r_i | r_j$ lorsque r_i est complètement disjointe de la région r_j

Toutes ces relations pouvant être combinées avec la relation de voisinage notée $\sim$, il est ainsi possible d'utiliser dix combinaisons, permettant de discriminer les différentes relations topologiques. La figure 4.14 illustre toutes les relations différentes que l'on peut rencontrer entre deux régions A et B.

Définition 87 Graphe d'association topologique

Soient $(G_i^1)_{i=0}^{k_1}$ *et* $(G_i^2)_{i=0}^{k_2}$ *deux hiérarchies de segmentations avec* $G_i = (V_i, E_i)$ *et les sommets* $v^1 \in {V_i}^1$, $w^1 \in {V_j}^1$, $v^2 \in {V_k}^2$, $w^2 \in {V_l}^2$.. *La paire de régions* (v^1, v^2) *est dite topologiquement consistante avec la paire* (w^1, w^2) *si la relation topologique existant entre* v^1 *et* w^1 *est la même que celle entre les sommets* v^2 *et* w^2 *. Le graphe d'association topologique de* $(G_i^1, {V_i}^1)_{i=0}^{k_1}$ *et* $(G_i^2, {V_i}^2)_{i=0}^{k_3}$ *est le graphe simple (voir Déf. 7)* $G_A = (V_A, E_A)$ *tel que :*

- $V_A = (\cup_{i=1}^{k_1} {V_i}^1) \times (\cup_{i=1}^{k_2} {V_i}^2)$
- $E_A = \{\{v, w\} \quad : \quad v \neq w \in V_A\}$, v *est topologiquement consistant avec* w

C'est en utilisant les relations topologiques illustrées sur la figure 4.14 qu'est construit le **graphe d'association topologique** à partir duquel est effectuée la recherche de la clique maximum. Pour détecter les cliques de poids maximum, Glantz et al utilisent une variante de l'algorithme PHB [MASSAR02], permettant de réduire la complexité de ce problème, originalement NP-complet (voir sections 4.1.3.e et 4.3).

La prise en compte des relations topologiques permet, comme pour les algorithmes présentés dans les sections 4.4.3 et 4.4.5 de réduire la complexité générale du problème en

guidant l'algorithme lors du processus d'appariement. Les résultats expérimentaux tendent à prouver que l'utilisation de hiérarchies de segmentations permet de réduire considérablement l'influence de sur-segmentations/sous-segmentations, en permettant à des régions appartenant à des niveaux différents d'être appariées, là où les algorithmes présentés dans les sections 4.4.3 et 4.4.5 ne possédaient qu'une représentation mono-échelle des partitions.

Par contre l'ajout de liens verticaux entre les graphes planaires détruit la structure planaire des deux graphes à apparier et fait perdre ainsi les nombreuses propriétés intéressantes inhérentes aux graphes planaires. De plus le graphe d'association topologique, obtenu à partir des deux pyramides, est une structure dont la taille est très importante. Pour pouvoir faire tourner l'algorithme dans des conditions acceptables, l'auteur réalise un compromis sur l'optimalité de la solution en limitant la construction de ce graphe aux 17 plus hauts niveaux de la pyramide.

CONTRIBUTION À L'APPARIEMENT DE HIÉRARCHIES

Sommaire

NOUS avons présenté, dans le chapitre précédent, le problème de l'appariement de graphes dans le domaine du traitement d'image, ainsi que des méthodes d'appariement de partitions mettant à profit certaines propriétés des graphes planaires.

Par rapport aux méthodes d'appariement de graphes présentées dans la section 4.1, les méthodes dédiées à l'appariement de partitions (section 4.4) s'emploient à faire diminuer la complexité de ce problème, à l'origine NP-complet, en prenant en compte l'orientation du plan d'une part (sections 4.4.3 et 4.4.2), mais également les relations topologiques existant entre les régions (section 4.4.5).

Néanmoins, ces approches ne permettent pas d'outrepasser les problèmes d'inconsistance inhérents à toutes les méthodes de segmentation. Différentes segmentations d'un même objet peuvent en effet amener à devoir mettre en correspondance une région d'une partition avec plusieurs régions dans l'autre partition. L'approche de Glantz [GLANTZ03] présentée dans la section 4.4.6 propose pour résoudre ce problème de ne plus considérer une seule partition pour chacune des images à apparier, mais une hiérarchie de partitions, permettant de prendre en compte différentes finesses de segmentation et les nombreux découpages possibles pour chacune des régions à mettre en correspondance. Néanmoins l'utilisation d'un graphe d'association, et l'ajout de liens verticaux entre les graphes planaires, correspondant aux niveaux de la pyramide, conduit à la perte de la planarité des graphes à apparier. On perd donc toutes les propriétés intéressantes des graphes planaires, susceptibles de faire chuter la complexité du problème.

Nous allons présenter dans ce chapitre une nouvelle approche au problème de l'appariement de régions, combinant les méthodes proposées par Llados [LLADOS01], Wang [CAIHUA95], Bunke [NEUHAU04] et Glantz [GLANTZ03] (voir section 4.1). En reprenant l'idée proposée par Glantz, nous allons dans un premier temps construire deux pyramides irrégulières à partir des deux images à apparier. La construction de ces hiérarchies se base sur les critères énergétiques définis dans le chapitre 3 et utilise les heuristiques de fusion que nous avons proposé dans les sections 3.2 et 3.3, dans le but d'obtenir des segmentations plus robustes aux variations locales.

Plutôt que de construire un graphe de taille conséquente comme Glantz, nous proposons d'initialiser l'appariement en cherchant dans les deux hiérarchies, un couple de régions dont les caractéristiques maximisent un critère de ressemblance. Ce critère est basé sur des propriétés aussi bien géométriques que photométriques. Nous allons ensuite confirmer cet appariement initial par la prise en compte des informations issues des voisinages.

Considérons deux pyramides combinatoires (voir Section 2.4.3.d) $P = (G_0, \ldots, G_N)$ et $P' = (G'_0, \ldots, G'_{N'})$. Chaque carte combinatoire G_i de P contient un ensemble de sommets. Nous noterons $\mathcal{V}_P$ l'union de tous les sommets de P définis entre les niveaux 0 et N. De la même façon l'ensemble des sommets appartenant à la pyramide P' est noté $\mathcal{V}_{P'}$. Le but de notre algorithme est de trouver deux sommets dans $\mathcal{V}_P$ et $\mathcal{V}_{P'}$ de caractéristiques similaires tels que l'on puisse définir deux coupes de P et P' où v et v' ont des voisinages similaires. Nous voulons donc déterminer le couple de sommets $(v, v') \in \mathcal{V}_P \times \mathcal{V}_{P'}$ pour lequel la distance entre les voisinages de v et v' est minimale :

$$(v, v') = \underset{(w,w')\in\mathcal{C}}{\text{argmin}} \Delta_N(w, w') \text{ avec} \quad (5.1)$$

$$\mathcal{C} = \{(w, w') \in \mathcal{V}_P \times \mathcal{V}_{P'} \,|F(w, w') = true\} \quad (5.2)$$

$\Delta_N(v, v')$ représente alors la distance minimale entre les voisinages de v et v' sur toutes les coupes de P et P' et $F(w, w')$ est une condition booléenne portant sur w et w'.

L'ensemble de couples des régions candidates à l'appariement étant de taille importante ($|\mathcal{V}_P| \times |\mathcal{V}_{P'}|$) , une première étape de filtrage va être effectuée. L'ensemble $\mathcal{C}$ représente notre ensemble de couples de régions candidates pour l'appariement. Avec la formulation que nous proposons, l'équation 5.2 est utilisée comme une étape de filtrage. Les avantages liés à cette étape sont doubles. Tout d'abord le critère F permet de garantir que deux sommets appariés auront nécessairement des caractéristiques similaires, indépendamment de la distance entre leurs voisinages. Elle permet également de réduire le nombre de couples de sommets pour lesquels l'équation 5.1 est évaluée. Cette dernière équation représente l'étape d'appariement et s'inspire des approches proposées par Neuhaus [NEUHAU04] et Llados [LLADÓS97, LLADOS01] (Section 4.4.2 et 4.4.3).
Les frontières des deux régions candidates sont mises en correspondance en utilisant un algorithme d'appariement de chaînes circulaires tirant parti des informations fournies par la hiérarchie. L'intérêt d'utiliser des hiérarchies tient dans le fait qu'elles vont nous permettre de considérer les différentes possibilités de réécriture des frontières pour chaque région au sein de sa hiérarchie en s'inspirant de l'approche proposée par Gdalyahu et al [GDALYA99A] (voir Section 4.4.4) mais en utilisant des systèmes de réécriture basés sur les informations fournies par les pyramides.

L'algorithme qui en résulte est un algorithme d'appariement de chaînes circulaires utilisant des règles de réécriture extraites des pyramides calculées pour chaque image (voir figure 5.1).

Nous allons dans la section 5.1 décrire en détail l'étape de filtrage permettant de réduire le nombre de sommets candidats à l'appariement puis dans la section 5.2 le processus permettant de calculer une distance basée sur les voisinages de deux régions.

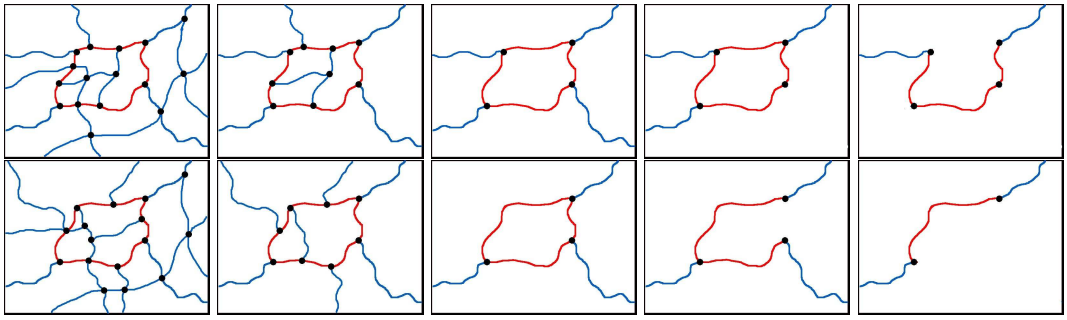


FIG. 5.1 – Deux exemples de hiérarchies de segmentation.

5.1 Filtrage

Pour réaliser le processus de filtrage, nous associons en premier lieu à chaque sommet v de l'ensemble $\mathcal{V}_P$ un vecteur $Feat(v)$ composé de M caractéristiques portant sur les régions. De nombreuses méthodes ont été proposées pour filtrer des bases de données de vecteurs. Toutefois, dans le cadre de notre application nous avons choisi de réutiliser les structures arborescentes des deux pyramides P et P'.

Supposons que le sommet v est défini au niveau $i \geq 1$ dans la pyramide P et considérons l'ensemble des sommets appartenant à la fenêtre de réduction du sommet v, $RW_i(v) = \{v_1, \ldots, v_p\}$ (voir Section 2.4.2), dont la contraction dans la carte G_{i-1} définit v au niveau i. Pour rappel $RW_i(v)$ est défini comme l'ensemble de sommets incidents à l'un des arbres du noyau de contraction, permettant de construire G_i à partir de la carte G_{i-1} (voir Déf. 56 dans la section 2.4.3.d). Nous utilisons de plus une fonction P, définie sur l'ensemble de sommets $\mathcal{V}_P$ telle que $P(w) = v$ pour tout sommet $w \in RW_i(v)$. Par convention tout sommet appartenant à la carte combinatoire de plus haut niveau dans la pyramide est considéré comme son propre père (i.e. $P(v_{top}) = v_{top}$).

Nous dirons qu'une coordonnée j de notre vecteur de caractéristiques est une *caractéristique croissante* si et seulement si pour tout sommet $v \in \mathcal{V}_P$, défini à un niveau $i \geq 1$ dans P, nous avons :

$$\forall w \in RW_i(v), \quad Feat_j(v) \geq Feat_j(w) \tag{5.3}$$

Nous pouvons noter que tout moment cumulé définit une caractéristique croissante sur la pyramide.

Supposons que la première composante $Feat_1(v)$ de notre vecteur de caractéristiques est croissante. La première étape consiste à réduire le nombre de couples de sommets candidats à l'appariement, en sélectionnant l'ensemble des couples $(v, v') \in \mathcal{V}_P \times \mathcal{V}_{P'}$ tels que :

$$\Delta_{F_1}(v, v') = |1 - \frac{Feat_1(v')}{Feat_1(v)}| \leq \epsilon \tag{5.4}$$

dans laquelle ϵ est un seuil défini par l'utilisateur.

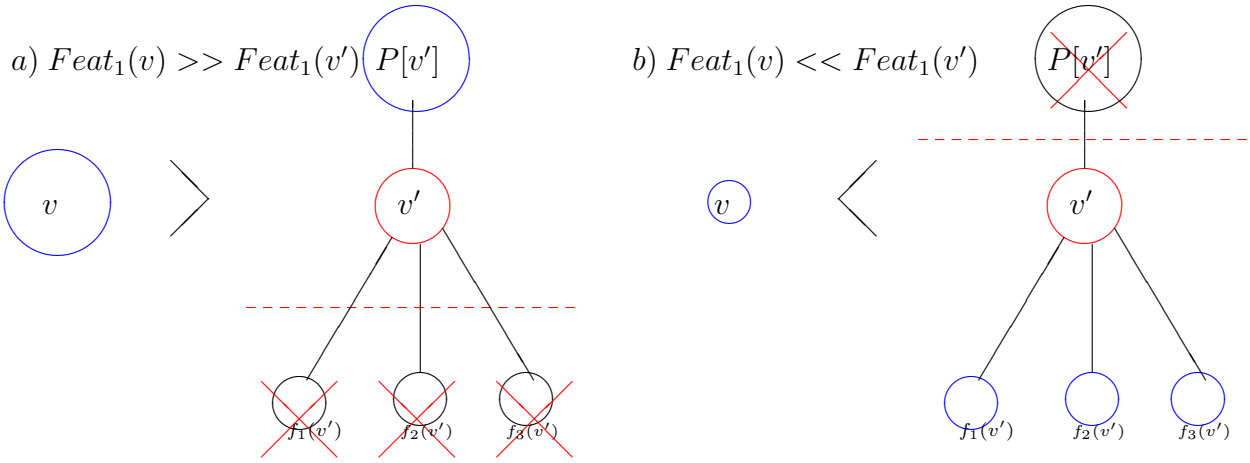


FIG. 5.2 – Différentes possibilités d'élagage. a) la recherche s'effectue récursivement sur les pères de v' ($P[v']$,$P[P[v']]$,...) . b) la recherche s'effectue récursivement sur les fils de v' ($f_i(v')$, $f_j(f_i(v)')$,...

Pour chacun des sommets $v \in \mathcal{V}_P$, cette première étape de filtrage revient à sélectionner tous les sommets v' de $\mathcal{V}_{P'}$ dont la première composante du vecteur de caractéristiques est comprise dans l'intervalle suivant :

$$\begin{aligned} Feat_1(v') &\in [(1-\epsilon)Feat_1(v), (1+\epsilon)Feat_1(v)] \text{ si } Feat_1(v) > 0 \\ Feat_1(v') &\in [(1+\epsilon)Feat_1(v), (1-\epsilon)Feat_1(v)] \text{ si } Feat_1(v) < 0 \end{aligned} \tag{5.5}$$

Pour tous les sommets v appartenant à la première pyramide, les sommets v'_i appartenant au même niveau dans la seconde seront comparés grâce à l'équation 5.4. En fonction du résultat la recherche sera propagée soit vers les fils des v'_i soit vers les pères de v'_i. Cette première étape de filtrage, définie par l'équation 5.4 permet de réduire le nombre de candidats potentiels à l'appariement en se basant uniquement sur une seule caractéristique croissante. Le fait d'utiliser un critère croissant permet ainsi de propager la recherche dans la pyramide, soit vers le père d'un sommet soit vers l'ensemble de sommets appartenant à sa fenêtre de réduction en fonction de l'équation 5.5.
La figure 5.2 illustre ce principe d'élagage dans le cas où la première caractéristique est toujours positive : dans la figure 5.2(a) $Feat_1(v') < (1-\epsilon)Feat_1(v)$, $Feat_1()$ étant une caractéristique croissante aucun des fils de v' ne peut satisfaire l'équation 5.5. La recherche se propagera donc récursivement vers les *pères* de v'. Inversement sur la figure 5.2(b) $Feat_1(v') > (1+\epsilon)Feat_1(v)$, $Feat_1(P[v'])$ est plus grand que $Feat_1(v)$ et ne peut donc satisfaire l'équation 5.5. La recherche s'effectuera dans ce cas récursivement sur l'ensemble des sommets de la fenêtre de réduction de v' (i.e. les *fils* de v').

Cette première sélection de candidats potentiels pour l'appariement est affinée par une seconde étape de filtrage utilisant le reste des caractéristiques disponibles pour chaque région. Les intervalles de valeurs relatifs aux différentes caractéristiques ayant une très grande variabilité, une étape de normalisation doit être effectuée avant de calculer la distance entre les caractéristiques des deux régions.

Étant données deux régions représentées par les sommets v et v', nous définissons la distance entre $Feat(v)$ et $Feat(v')$ comme la norme infinie d'un vecteur f défini par :

$$\forall i \in \{1, \ldots, M\} \quad f_i(v, v') = 1 - \frac{\min(Feat_i(v), Feat_i(v'))}{\max(Feat_i(v), Feat_i(v'))} \tag{5.6}$$

Finalement les sommets appartenant à l'ensemble $\mathcal{C}$, correspondant aux couples de sommets ayant passé l'étape de filtrage (equation 5.2), doivent respecter les critères booléens suivant :

$$F(w, w') = \Delta_{F_1}(v, v') \leq \epsilon \tag{5.7}$$
$$\|f\|_\infty \leq \epsilon. \tag{5.8}$$

Ces deux conditions étant testées de façon séquentielle.

Proposition 3 *Si la première caractéristique $Feat_1(v)$ est positive, alors :*

$$\Delta_{F_1}(v, v') \leq \epsilon : f_1(v, v') \leq \epsilon \tag{5.9}$$

La démonstration de cette proposition est donnée en annexe (section A.2).

La proposition 3 montre que lorsque $Feat_1()$ est une caractéristique positive, notre première étape de filtrage $\Delta_{F_1}(v, v') \leq \epsilon$ peut être interprétée comme une restriction du second test ($\|f\|_\infty \leq \epsilon$) à la première composante du vecteur f. Une telle interprétation ne tient plus si l'on utilise une norme Euclidienne à la place de la norme infinie. Nous avons néanmoins observé lors d'expérimentations, que le fait d'utiliser une norme Euclidienne peut s'avérer préférable si l'on souhaite obtenir un phénomène de compensation entre les différentes composantes f_i.

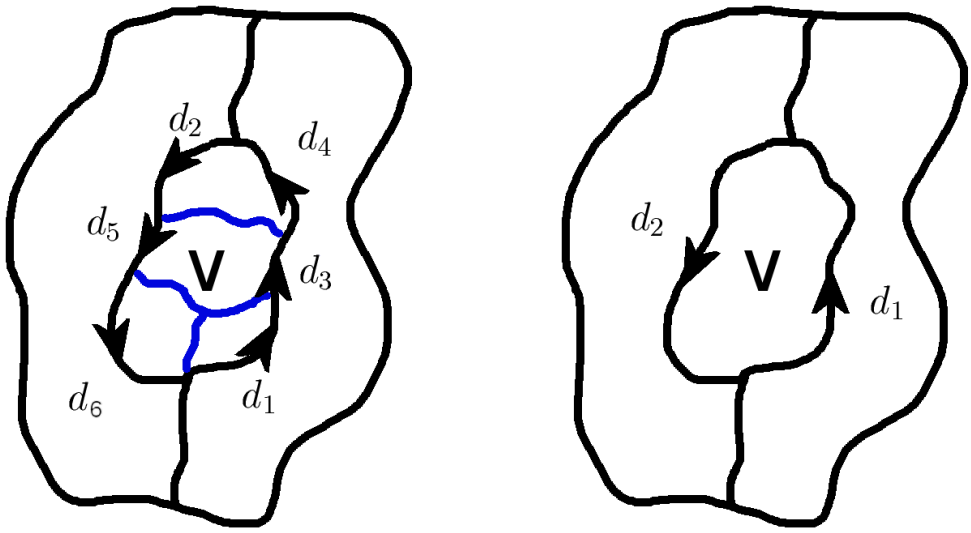
5.2 Distance hiérarchique entre les voisinages

Nous allons, dans cette section, définir une distance hiérarchique permettant de comparer les voisinages de deux régions en prenant en compte les différents voisinages possibles d'une région dans la pyramide.

Soit une pyramide P, considérons un sommet v appartenant à l'ensemble de sommets $\mathcal{V}_P$. Nous noterons l_v le niveau maximum pour lequel v survit dans la pyramide P. Nous représenterons par $\sigma^*_{l_v}(d) = (d_1, \ldots, d_q)$ le σ-cycle de G_{l_v} associé au sommet v. Le plongement de la frontière de v dans la carte de base G_0, noté B_v est défini comme la concaténation des séquences de brins frontières $(SBD_{l_v}(d_j))_{j \in \{1,\ldots,q\}}$ (Déf. 63).

Comme tous les brins d'une séquence de brins frontières appartiennent à la carte de base $G_0 = (\boldsymbol{\mathcal{D}}_0, \sigma_0, \alpha_0)$, chaque sommet de la hiérarchie peut ainsi être considéré comme un mot construit sur l'alphabet $\boldsymbol{\mathcal{D}}_0$. Le voisinage de v dans G_0 est défini comme l'ensemble des sommets de G_0 adjacents à l'éventuelle sur-segmentation de v. Cet ensemble noté $N^0(v)$ peut être formalisé de la façon suivante :

$$N^0(v) = \{\sigma^*_0(\alpha_0(d)), d \in B_v\} \tag{5.10}$$



(a) niveau 0 (b) niveau l_v

FIG. 5.3 – La régions v présente sur la partition (b) à un niveau n est sur segmentée en 4 régions distinctes sur la carte du niveau de base (a)

La figure 5.3 illustre ces notions. Une région v définie à un niveau l_v (Fig. 5.3(b)) se trouve être sursegmentée au niveau de base (Fig. 5.3(a)). Le plongement B_v de sa frontière au niveau de base est égal à $B_v = d_1 \dots d_6$. Le voisinage $N^0(v)$ est défini par $N^0(v) = \{\sigma_0^*(\alpha_0(d_1)), \dots, \sigma_0^*(\alpha_0(d_6))\}$.

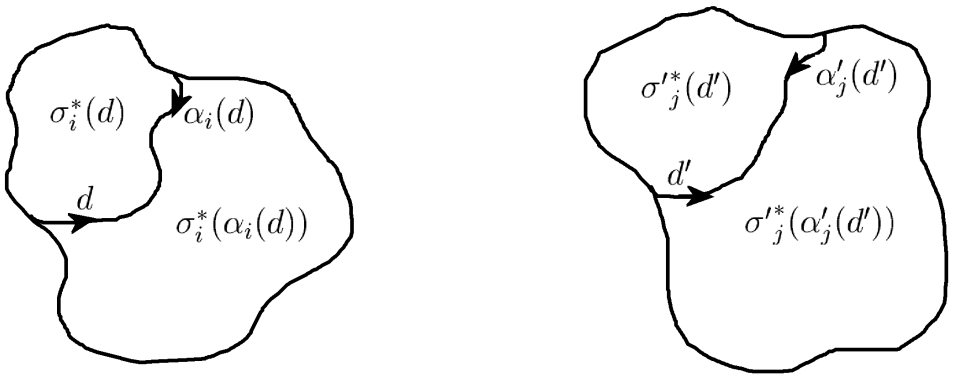
Si l'on considère maintenant deux pyramides P et P' ainsi que deux sommets $(v, v') \in \mathcal{V}_P \times \mathcal{V}_{P'}$ avec leurs plongements respectifs définis par $B_v = d_1 \dots, d_n$ et $B_{v'} = d'_1 \dots . d'_{n'}$. En utilisant l'orientation naturelle des voisinages $N^0(v)$ et $N^0(v')$ dans les cartes G_0 and G'_0, nous définissons la distance de voisinage entre les régions v et v' au niveau 0 comme la distance d'édition entre B_v et $B_{v'}$.

Si q représente le nombre de symboles mis en correspondance entre B_v et $B_{v'}$, le coût total correspondant à l'appariement entre les sommets v et v' est défini par :

$$\Delta(B_v, B_{v'}) = \sum_{i=0}^{q} \delta(d_{\phi_1(i)}, d'_{\phi_2(i)}) + (n + n' - 2q)K \tag{5.11}$$

avec $\delta(,)$ représentant une fonction de distance entre les brins et K le coût par défaut affecté à une opération de suppression effectuée indifféremment dans B_v ou $B_{v'}$. Les valeurs de $\phi_1(i)$ et $\phi_2(i)$ correspondent aux index des i^{emes} appariement de brins dans B_v et $B_{v'}$.

Comme nous l'avons vu dans la section 2.3.3, chaque brin appartenant à une carte combinatoire peut être interprété de deux façon distinctes : soit comme la représentation de l'adjacence entre deux régions, soit comme une frontière entre ces deux mêmes régions. Si nous considérons les deux brins $d \in G_i$ et $d' \in G'_j$ dans les pyramides P et P', la distance $\delta(d, d')$ entre ces deux brins peut prendre en compte à la fois les caractéristiques :



(a) (b)

FIG. 5.4 – Illustration des notations utilisées pour décrire des régions à des niveaux i et j, de part et d'autre du brin d (a) et d' (b).

- des sommets $\{\sigma_i^*(d), \sigma_i^*(\alpha_i(d)), \sigma_j'^*(d'),\ \sigma_j'^*(\alpha_j(d'))\}$ qui correspondent aux régions de part et d'autre des ces brins,
- des frontières associées aux brins d et d' aux niveaux i et j.

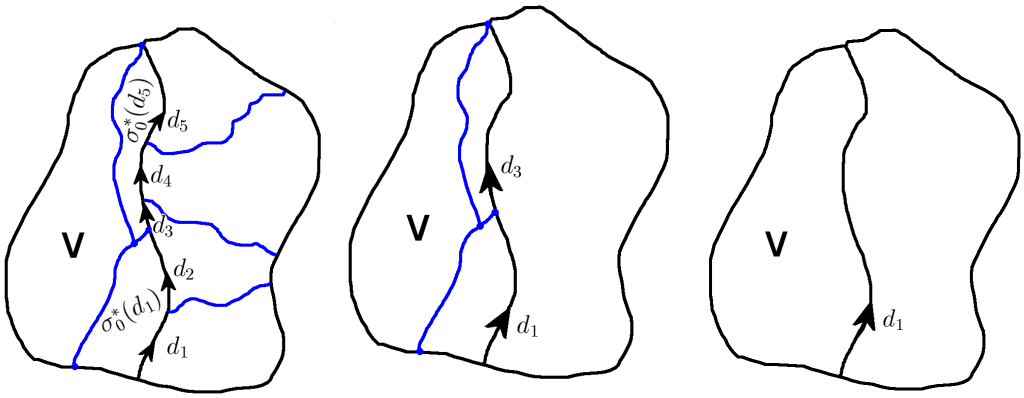
La figure 5.4 illustre les notations utilisées dans cette section pour décrire les régions ainsi que les frontières associées.
Dans notre schéma d'appariement, la distance entre les caractéristiques des sommets v et v' ayant déjà été prise en compte lors de l'étape de filtrage, nous définissons la distance $\delta(d, d')$ en fonction des caractéristiques des sommets $\sigma_i^*(\alpha_i(d))$, $\sigma_j'^*(\alpha_j(d'))$ ainsi que des caractéristiques associées aux deux frontières codées par les brins d et d' aux niveaux i et j.

5.2.1 Coupes et séquences de brins frontières

Tous les brins composant les séquences B_v et $B_{v'}$ sont définis sur G_0 et G'_0. De ce fait, la distance $\delta(.,.)$ de l'équation 5.11 est calculée au niveau de base de la pyramide et la comparaison entre les voisinages des sommets v et v' est effectuée dans les cartes G_0 et G'_0. Cette approche est comparable à celle proposées par Bunke [NEUHAU04] et Llados [LLADÓS97, LLADOS01] (Sections 4.4.2 et 4.4.3) et peut se résoudre en utilisant un algorithme d'appariement de chaînes circulaires [MAES90, BUNKE93A, LLADOS01, PERIS02, MOLLIN02, ROBLES02].

Néanmoins, l'inconsistance inhérente à tout algorithme de segmentation fait que les voisinages de v et v' ont peu de chance de se correspondre si l'on considère uniquement les partitions de base encodées par les cartes G_0 et G'_0.
Pour deux pyramides P et P', construites respectivement à partir de G_0 et G'_0, nous pouvons considérer l'ensemble des coupes dans P (resp. P') définies sous v (resp. v'), c'est

à dire à un niveau inférieur au niveau maximum pour lequel le sommet v (resp. v') survit. Dans ce cas de figure, le sommet v (resp. v') représente soit une unique région ou bien il se retrouve découpé en plusieurs sous régions. L'idée de base, sur laquelle repose le développement qui va suivre, part du principe que si v et v' correspondent à une même région, alors il devrait y avoir au moins une coupe dans chaque pyramide pour lesquelles les voisinages des régions v et v' sont isomorphes.



(a) niveau 0 (b) niveau l_1 (c) niveau l_2

FIG. 5.5 – Représentation d'une frontière à différents niveaux d'une pyramide.(a) représente la carte combinatoire de base de la pyramide réduite en l_1 elle-même réduite en l_2 (i.e $l_1 \leq l_2$).

Considérons une sous séquence maximale $d_l \ldots . d_m$ de B_v telle que l'ensemble des sommets $\{\sigma_0^*(\alpha_0(d_l)), \ldots, \sigma_0^*(\alpha_0(d_m))\}$ est fusionné en un seul sommet à un niveau $l_1 \leq l_v$. Tous les brins codant les relations d'adjacence entre $\{\sigma_0^*(\alpha_0(d_l)), \ldots, \sigma_0^*(\alpha_0(d_m))\}$ doivent alors être soit contractés soit retirés, en tant que boucles vides (Déf. 17, section 2.1), au niveau l_1. De la même façon, l'ensemble des sommets $\{\sigma_0^*(d_l), \ldots, \sigma_0^*(d_m)\}$ définissent une sur-segmentation du sommet v et doivent être fusionnés en un unique sommet à un niveau $l_2 \leq l_v$. Si l'on considère désormais la séquence $d_l \ldots . d_m$ au niveau $L = \max(l_1, l_2)$, les brins $d_l \ldots . d_m$ représentent une séquence de morceaux d'une frontière existant entre deux régions et peuvent alors être interprétés comme des arêtes doubles [BRUN02] (Déf. 18 section 2.1). Ces brins appartiennent donc à la séquence de brins frontières $SBD_L(d_k)$ d'un brin $d_k \in B_v$.
Inversement, une séquence de brins frontières $SBD_L(d_k) = d_l \ldots . d_m$ avec $L \leq l_v$ et $d_k \in B_v$ code une séquence d'arêtes doubles, pouvant être créées uniquement par la fusion des sommets $\{\sigma_0^*(d_l), \ldots, \sigma_0^*(d_m)\}$ et $\{\sigma_0^*(\alpha_0(d_l)), \ldots, \sigma_0^*(\alpha_0(d_m))\}$. La figure 5.5 illustre ces notations en présentant une région v à trois niveaux de la pyramide. La séquence de brins frontières séparant la région v de la région qui lui est adjacente est codée par les brins $d_1 \ldots . d_5$ au niveau 0, par d_1, d_3 au niveau l_1 et enfin par l'unique brin d_1 au niveau l_2.

Une séquence de brins frontières nous permet donc de retrouver l'ensemble des opérations de fusion ayant eu lieu entre les sommets composant le voisinage $N^0(v)$ à un niveau inférieur à l_v.
Du fait que deux coupes produisant les mêmes opérations de fusion sur $N^0(v)$ amènent à la même configuration locale pour le voisinage du sommet v, la séquence de brins frontières

$SBD_i(d)$ avec $i \leq l_v$ et $d \in B_v$ nous permet de retrouver tous les voisinages possibles pour le sommet v, produits par les différentes coupes dans la pyramide P.

5.2.2 Algorithme de réécriture

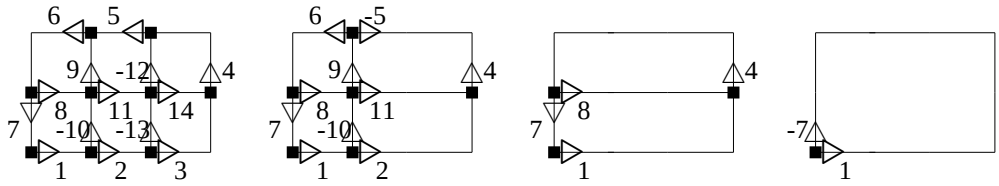


FIG. 5.6 – Exemple de constructon d'une pyramide sur une grille 3×2

Pour retrouver l'ensemble des voisinages d'un sommet v, survivant jusqu'à un niveau l_v dans une pyramide P nous pouvons utiliser une solution triviale qui consiste à partir du niveau l_v et de descendre successivement les niveaux en calculant $\sigma^*(d)$ pour chacun d'eux. Cette solution est néanmoins extrêmement coûteuse puisqu'elle nécessite de reconstruire chacun des niveaux de la pyramide qui sont codés implicitement.

Algorithme 3 Calcul la séquence de brins frontières d'un brin b

entrée : un brin b **, une carte combinatoire** $G_i = (D_i, \sigma_i, \alpha_i)$ **du niveau courant** i **et la carte** $G_{base} = (D_{base}, \sigma_{base}, \alpha_{base})$ **correspondant à la base de la pyramide**
sortie : séquence de brins frontière d'un brin b
$Reecritures := \emptyset$
$Reecriture[base] := \emptyset$
$niveau := \emptyset$
if (giveLevel(α_{base}(b))==i) **then**
  renvoyer b
end if
$b_{courant} := b$
while $\alpha_{base}(b_{courant}) \neq \alpha_i(b)$ **do**
  $b_{courant} := \sigma_{base}(b)$
  $level := level(b_{courant})$
  while ((level < i) et (etat(level) $\in \{contraction, boucleVide\}$)) **do**
    $b_{courant} := \sigma_{base}(\alpha_{base}(b))$
    $level := giveLevel(b_{courant})$
  end while
  inserer $level$ dans $niveau$
  ajouter $b_{courant}$ à $Reecriture[base]$
end while
construireRéécriture()

Afin de calculer ces voisinages d'une manière plus efficace, nous avons défini un premier algorithme permettant de retrouver les réécritures d'une frontière se déroulant en deux temps.

- un premier algorithme (Algo. 3) permet de calculer la séquence de brins frontières (au niveau de base).
- à partir de la séquence de brins frontières nous calculons les réécritures pour chaque niveaux grâce à la fonction *construireRéécriture* (Algo. 4).

Nous utilisons dans cet algorithme la fonction $niveau$ qui retourne, pour tout brin b, le niveau auquel il disparait dans la pyramide. Nous disposons également d'une fonction $etat$ permettant de savoir si un brin a disparu suite à une étape de suppression ou lors d'une opération de contraction.

Cet algorithme (Algo. 3) va donc dans un premier temps retrouver la réécriture d'un brin au niveau de base de la pyramide. En parallèle à la construction de la séquence de brins frontières, il construit une liste des différents niveaux correspondant aux différentes réécritures. Il fait ensuite appelle à la fonction *construireRéécriture*, qui se chargera de retrouver toutes les réécritures possibles de ce brins aux différents niveaux de la pyramide, mémorisés au préalable pour éviter de calculer plusieurs fois une même réécriture.
La fonction *construireRéécriture* (Algo. 4) va ainsi construire l'ensembles des réécritures, en parcourant uniquement la séquence de brins correspondant à la réécriture de niveau précédent.

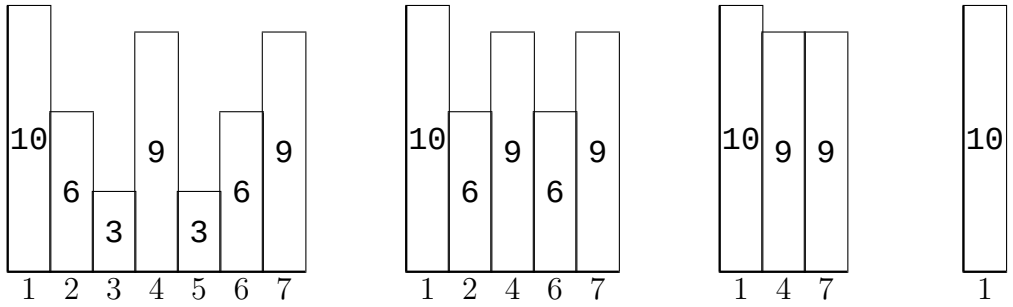
Algorithme 4 fonction permettant de reconstruire l'ensemble des réécritures possibles d'un brin, à partir de son écriture au niveau de base de la pyramide

fonction construireRéecriture
taille=longueur(niveau)
for n=1 à (taille-1) **do**
  for tout brin d dans $Reecritures[niveau[n-1]]$ **do**
    if giveLevel(d) $<$ n **then**
      ajouter d dans $Reecritures[niveau[n]]$
    end if
  end for
end for
renvoyer $Reecritures$

Prenons par exemple l'unique région présente dans la carte de la pyramide (fig. 5.6.d), codée par le brin 1. L'algorithme 3 va dans un premier temps nous fournir sa réécriture au niveau de base, à savoir $(1, 2, 3, 4, 5, 6, 7)$, ainsi qu'un tableau avec les niveaux dans lesquels des réécritures seront effectuées, pour cet exemple $niveau = (0, 3, 6, 9)$.

index	0	1	2	3
niveau	0	3	6	9

TAB. 5.1 – tableau $niveau$ pour l'exemple de la fig. 5.6



a) niveau 0 b) niveau 3 c) niveau 6 d) niveau 9

FIG. 5.7 – Les différentes réécritures possibles d'un contour de région correspondant aux niveaux donnés en exemple dans la figure. 5.6. Pour chaque brin, le niveau auquel il survit est indiqué dans la barre correspondante.

On peut également calculer l'ensemble des réécritures de chaque brin à partir des séquences de connexion (Déf 61) défini par l'algorithme 5.9 [BRUN02]. Notre calcul de réécritures est basé sur le fait que l'ensemble des brins composant une réécriture d'un brin b au niveau j est compris entre b et $\sigma_j(b)$ dans la séquence B_v. On peut donc modifier l'algorithme 5.9 sur la ligne notée par $\star$ de façon à parcourir l'ensemble des brins frontières compris entre $\sigma_{j-1}(prec[j])$ et b'. Cet ensemble de brins frontière peut être stocké dans un tableau auxiliaire mis à jour au fur et à mesure du parcours de la séquence de connexion. Par exemple lorsque l'algorithme 5.9 rencontrera le brin 7 celui ci va initialiser $\sigma_9(4)$ à 7. Les deux réécritures possibles du brin 4 sont alors définies par $\{4, 5, 6\}$ et $\{4, 6\}$ (figure 5.8). Ce second algorithme bien que présentant certaines propriétés intéressantes n'a pas été pleinement testé faute de temps.

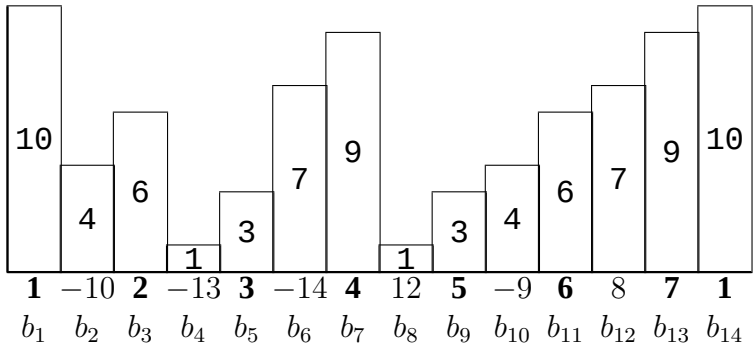


FIG. 5.8 – La séquence de connexion du brin 1. Les brins frontières sont représentés en gras.

5.2.3 Appariement hiérarchique et circulaire de chaînes

Pour toute séquence de brins frontières $SBD_i(d_j) = d_j.\ldots.d_k$ avec $i \leq l_v$ et $d_j \in B_v$ nous considérons la règle de réécriture suivante :

$$\psi^i_{j,k} : d_j.\ldots.d_k \mapsto d_j \tag{5.12}$$

ALGORITHME CONSTRUIRERÉECRITURE(int i,brin b) :

entrée : la carte du niveau de base $G_0 = (D_0, \sigma_0, \alpha_0)$, un brin b et son niveau maximum i.

sortie : fournit les différentes réécritures possibles d'un brin en mettant à jour les fonctions σ et φ pour tous les brins à chaque niveau permettant d'obtenir les séquences de brins frontières pour chaque niveau.

brin b'
brin prec[i+1]
for (j=1 à min($niveau(b)$,i+1) **do**
  prec[j]=b
end for
while $niveau(b') \leq i$ **do**
  if $(etat(niveau(b') ==$ Contracté$)$ ou
  $(niveau(b') > i$ et $etat(i) ==$ Contracté$)$ **then**
    $b' = \varphi(b')$
  else
    $b' = \sigma(b')$
  end if
  for (j=1 à min($niveau(b')$,i+1) **do**
    if $(etat(niveau(prec[j]) ==$ Contracté$)$ ou
    $(niveau(prec[j]) > i$ et $etat(i) ==$ Contracté$)$ **then**
      $\varphi_{j-1}(prec[j]) = b'$
    else
      $\sigma_{j-1}(prec[j]) = b' \quad \star$
    end if
  end for
  for (j=1 à min($niveau(b')$,i+1) **do**
    prec[j]=b'
  end for
end while

FIG. 5.9 – Algorithme permettant de retrouver l'ensemble des réécritures possibles d'une région, à partir de sa séquence de connexion (au niveau de base de la pyramide).

qui consiste à remplacer dans B_v, la séquence de brins $d_j \ldots, d_k$, définie dans la carte de base G_0, par un unique brin d_j appartenant à la carte de niveau supérieur G_i. Pour éviter toute confusion entre le brin d_j appartenant à la carte de base G_0 et celui appartenant à G_i nous allons utiliser la même notation pour le nom de la règle $\psi^i_{j,k}$ et son résultat d_j dans G_i. Nous avons ainsi :

$$\psi^i_{j,k} : d_j.\ldots.d_k \mapsto \psi^i_{j,k} \tag{5.13}$$

D'un point de vue purement géométrique, la frontière orientée définie par le brin $\psi^i_{j,k}$ peut être définie comme la concaténation des éléments de frontières représentés par $d_j \ldots, d_k$ au niveau 0.

Soit Σ l'ensemble de règles $\psi^i_{j,k}$ définies sur B_v et $\Sigma(B_v)$ l'ensemble des réécritures possibles de la frontière B_v par des règles $\psi^i_{j,k}$ (calculées dans la section 5.2.2). Considérons de la même façon l'ensemble des réécritures possibles de $B_{v'}$, noté $\Sigma'(B_{v'})$.
Les deux ensembles $\Sigma(B_v)$ et $\Sigma'(B_{v'})$ représentent alors les différents voisinage possibles de v et v' pour les différentes coupes dans les pyramides P et P'.

Nous définissons alors la distance entre les voisinages de v et v' par l'expression suivante :

$$\Delta_N(v, v') = \min_{(m,m') \in Rot(\Sigma(B_v)) \times \Sigma'(B_{v'})} \Delta(m, m') \tag{5.14}$$

dans laquelle Δ est une distance d'édition similaire à celle définie par l'équation 5.11. $Rot(\Sigma(B_v))$ représente l'ensemble des permutation circulaires possibles sur l'ensemble des chaînes contenues dans $\Sigma(B_v)$.

Pour une rotation donnée, une telle distance peut être calculée efficacement [GDALYA99B] en utilisant les équations récursives suivantes :

$$\Delta(0, 0) = 0 \tag{5.15}$$

$$\Delta(i, j) = \min \begin{pmatrix} \Delta(i-1, j-1) + \delta(d_i, d_j), \\ \min\limits_{\psi^l_{k,i} \in \Sigma, \psi'^{l'}_{k',j} \in \Sigma'} \Delta(k-1, k'-1) + \delta(\psi^l_{k,i}, \psi'^{l'}_{k',j}), \\ \Delta(i-1, j) + K, \\ \Delta(i, j-1) + K \end{pmatrix} \tag{5.16}$$

- $\Delta(i-1, j-1) + \delta(d_i, d_j)$ représente un prolongement d'un appariement en rajoutant le couple $(d_i \leftrightarrow d_j)$.
- $\min\limits_{\psi^l_{k,i} \in \Sigma, \psi'^{l'}_{k',j} \in \Sigma'} \Delta(k-1, k'-1) + \delta(\psi^l_{k,i}, \psi'^{l'}_{k',j})$ permet de calculer la réécriture fournissant le coût d'appariement minimum alors que
- $\begin{cases} \Delta(i-1, j) + K, \\ \Delta(i, j-1) + K \end{cases}$ représente les coûts de suppression d'un élément pour chacune des deux chaînes.

Dans l'équation 5.16, $\psi^{l}_{k,i}$ et ${\psi'}^{l'}_{k',j}$ représentent deux règles de réécriture définies sur les alphabets Σ et Σ'. Les index k et k' sont respectivement plus petits que i et j et les niveaux l et l' doivent être respectivement inférieurs aux niveaux l_v et $l_{v'}$. Notons également que la distance $\delta(d_i, d_j)$ dans l'équation 5.16 est calculée à partir de la carte de base au niveau 0.

Dans un premier temps nous allons détailler les étapes majeures de cet algorithme d'appariement hiérarchique, puis dans la section 5.2.5 nous allons le faire se dérouler sur un exemple simple.

A partir de deux séquences de brins, ainsi que les réécritures pour chacun de ces brins, nous désirons apparier ces deux séquences en minimisant les différents coûts occasionnés, par les différentes opérations d'édition.

Algorithme 5 Calcule un tableau $R_{i,j}$ des coûts d'appariement des deux séquences passées en arguments

entrée : deux séquences de brins A et B avec les réécritures possibles pour chacun des brins des deux séquences
sortie : le tableau de coûts $R_{i,j}$
for tout brin b_i dans A **do**
  for tout brin b_j dans B **do**
    minL = calculReecLigne(b_i, b_j)
    minC = calculReecColonne(b_i, b_j)
    minLC = calculReecLigneColonne(b_i, b_j)
    minCell = calculCellule(b_i, b_j)
    $\Delta[i, j]$ = MIN(minL,minC,minLC,minCell)
  end for
end for
renvoyer R

L'algorithme 5 présente l'algorithme général de calcul d'une matrice de coût $\Delta[n, m]$, permettant ensuite de retrouver l'appariement optimal, entre les deux séquences de brins passées en paramètre, en se basant sur l'équation 5.16.
A chaque étape, cet algorithme va calculer le coût minimum necéssaire pour apparier deux brins b_i et b_j en considérant

- les réécritures du brin b_i (calculReecLigne(b_i, b_j))
- les réécritures du brin b_j (calculReecColonne(b_i, b_j))
- les réécritures combinées du brin b_i et du brin b_j (calculReecLigneColonne(b_i, b_j))
- la substitution du brin b_i par le brin b_j (calculCellule(b_i, b_j))

La fonction $calculReecLigne$ retourne le coût minimum d'appariement entre les deux brins b_i et b_j en prenant en compte les différentes réécritures possibles du brin b_i et en y intégrant un coût de transformation (via réécriture). Ce coût peut être défini comme assez faible du fait que ces réécritures existent dans la pyramide, et qu'elles n'impliquent donc pas de modifications structurelles.

FONCTION CALCULREECLIGNE(b_i,b_j) :
mini = ∞

for toutes les réécritures rb_i possibles du brin b_i **do**
if $\Delta[rb_i, b_j]$ + coûtDeTransformation < mini **then**
mini = $\Delta[rb_i, j]$ + coûtDeTransformation
end if
end for
renvoyer mini

De même, la fonction $calculReecColonne$ va retourner le coût minimum de mise en correspondance des brins b_i et b_j en prenant en compte les différentes réécritures possibles du brin b_j .

FONCTION CALCULREECCOLONNE(b_i,b_j) :
mini = ∞
for toutes les réécritures rb_j possibles du brin b_j **do**
if $\Delta[i, rb_j]$ +coûtDeTransformation < mini **then**
mini = $\Delta[i, rb_j]$ + coûtDeTransformation
end if
end for
renvoyer mini

La fonction $calculReecLigneColonne$ permet, quant à elle, de prendre en compte les réécritures pour les deux chaînes en même temps.

FONCTION CALCULREECLIGNECOLONNE(b_i,b_j) :
mini = ∞
for toutes les réécritures rb_i possibles du brin b_i **do**
for toutes les réécritures rb_j possibles du brin b_j **do**
if $\Delta[rb_i, rb_j]$+coûtDeTransformation < mini **then**
mini = $\Delta[rb_i, rb_j]$ + coûtDeTransformation
end if
end for
end for
renvoyer mini

Enfin, la fonction $calculCellule$ va retourner le coût minimum d'appariement des brins b_i et b_j, en prenant en compte les suppressions possibles (en y affectant un coût important) et également l'extension de l'appariement précédent en ajoutant simplement un coût de substitution calculé sur la diagonale.

FONCTION CALCULCELLULE(b_i,b_j) :
minC = $\Delta[i, j-1]$ + CoutSuppression
minL = $\Delta[i-1, j]$ + CoutSuppression
minDiag = $\Delta[i-1, j-1]$ + CoutDiag(i,j)

renvoyer MIN(miniC,miniL,miniDiag)

La fonction $calculCellule$ nécessite l'utilisation de la fonction $CoutDiag(i,j)$ donnant le coût nécessaire pour la substitution du brin b_i avec le brin b_j. Ce coût peut être calculé avec toutes les informations que l'on possède sur nos brins (longueur du contour, gradient le long du contour, couleurs, moments ...).

En parallèle à la construction du tableau de coûts, un tableau $Retour$ (Figure 5.16) permettant de retrouver les différentes opérations effectuées est construit. Ce dernier va ainsi nous permettre de mettre en évidence le meilleur appariement possible entre les deux séquences de brins.

Pour retrouver le chemin de l'appariement optimal, précédemment calculé, nous avons codé les opérations d'édition de la façon suivante :

- 0.5 pour une mise en correspondance
- n pour une réécriture en colonne, transformant une séquence de n brins en 1 brin
- $-n$ pour une réécriture en ligne, transformant une séquence de n brins en 1 brin
- 0 pour une suppression

Il suffit ensuite, en partant de la dernière case (n,m) de la matrice de coût, de remonter en effectuant les opérations mémorisées, jusqu'à l'obtention de l'appariement global.

5.2.4 Analogie avec l'algorithme de plus court chemin

En étudiant la construction de la matrice de coût définie dans la section 5.2, nous pouvons noter une analogie entre l'algorithme 5 que nous proposons et l'algorithme de plus court chemin (Dijkstra). Ce dernier permet de trouver un chemin de longueur minimale entre deux sommets d'un graphe pondéré orienté ou non. Cet algorithme est présenté en détail en annexe dans la section A.4.

Soient deux chaînes $A^0 = \{a_0^0, \ldots, a_n^0\}$ et $B^0 = \{b_0^0, \ldots, b_m^0\}$ définies sur des alphabets Σ et Σ', munies de règles de réécriture $\psi_{k,i}^l$ et $\psi'^{l'}_{k',j}$. Comme nous l'avons vu dans la section précédente la matrice de coût va être calculée, en considérant les appariements triviaux, mettant en correspondance des éléments du niveau de base, puis ceux nécessitant des réécritures.

Nous allons définir un graphe $G = (V,E)$ à partir duquel nous allons calculer un chemin de poids minimum. Nous appelerons D le sommet de départ dans le graphe G et nous noterons $(a_p^l, b_{p'}^{l'})$ un sommet du graphe. Ce sommet codera l'appariement du brin a_p^l (p-ième élément du niveau l) avec $b_{p'}^{l'}$.

L'ensemble E des arêtes du graphe G est constitué par les arêtes :

- incidentes au sommet de départ D
  - $[D, (a_0^0, b_0^0)]$ correspondant à l'appariement des deux premiers élément de chacunes des chaînes à mettre en correspondance
  - $[D, (a_0^l, b_0^{l'})]$ si $\exists \psi_{0,k}^l, \psi_{0,k'}^{l'}$ codant les réécritures de brins $a_0^0, \ldots, a_k^0$ en a_0^l ainsi que $b_0^0, \ldots, a_{k'}^0$ en $b_0^{l'}$.
- ainsi que les autres arêtes :

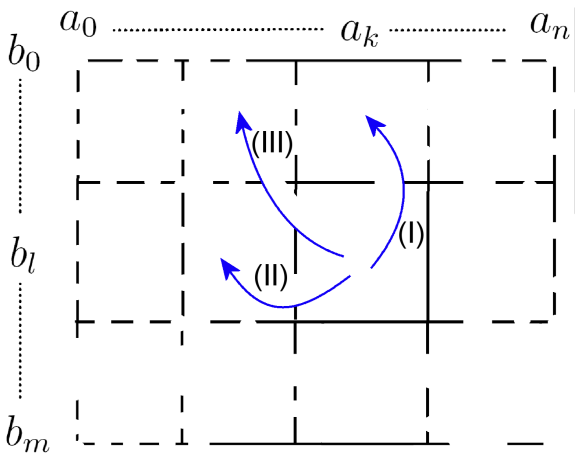


FIG. 5.10 – Les opérations de suppressions (I) et (III) ainsi que l'opération de substitution (II) représentée dans la matrice de coût.

– $[(a_k^i, b_l^j, (a_{k'}^{l'}, b_{l'}^{j'})]$ si

$$\begin{cases} (I) & k' = k+1, l' = l & \text{opération de suppression} \\ (II) & k' = k, l' = l+1 & \text{opération de suppression} \\ (III) & k' = k+1, l' = l+1 & \text{opération de substitution} \\ (IV) & k' \geq k, l' \geq l \text{ si } \exists \psi_{k,k'}^i, \psi_{l,l'}^j & \text{opération de réécriture} \end{cases}$$

Ces arêtes vont ainsi correspondre aux différentes opérations d'édition que nous avons défini dans la section 5.2. La figure 5.11 représente les opérations de suppressions représentées par les arêtes (I) et (II) dans le graphe G alors que l'opération de substitution correspondant à la mise en correspondance de deux éléments venant élargir l'appariement courant correspond à une arête de type (III). Les arêtes de type (IV) ne sont représentés mais relient sur la figure 5.11 la case (k, l) à des case de la sous matrice $0 \ldots k \times 0 \ldots l$.

L'ensemble des sommets V' avec lesquels nous allons effectivement travaillé peut être défini comme la restriction de l'ensemble des sommets :

$$V = \{(a_k^i, b_l^j), \quad k \in \{0, \ldots, N\}, l \in \{0, \ldots, M\}\}$$

à la composante connexe $CC(D)$ de G contenant D.

Le graphe G'=(V',E) est connexe et orienté et par construction toute arête $[(a_k^i, b_l^j), (a_{k'}^{i'}, b_{l'}^{j'})]$ satisfait $k' + l' > k + l$. Le graphe résultant est donc acyclique. Chaque sommet de ce graphe correspond à l'appariement d'un élément de A (ou une réécriture) avec un élément de B (ou une réécriture) alors que les arêtes représentent le coût nécessaire pour cette mise en correspondance. Le fait de calculer le plus court chemin menant du sommet D à l'un des sommets de ce graphe n'ayant pas d'arêtes sortante est équivalent au calcul de notre matrice de coût. Chaque élément $M_{i,j}$ de la matrice de coût représente ainsi la distance minimale permettant d'apparier la sous-chaîne $a_0, \ldots, a_i$ avec $b_0, \ldots, b_j$ en prenant en compte les réécritures.

La figure 5.12 illustre le type de graphe acyclique obtenu lors de l'appariement des deux chaînes représentées sur la figure 5.11.

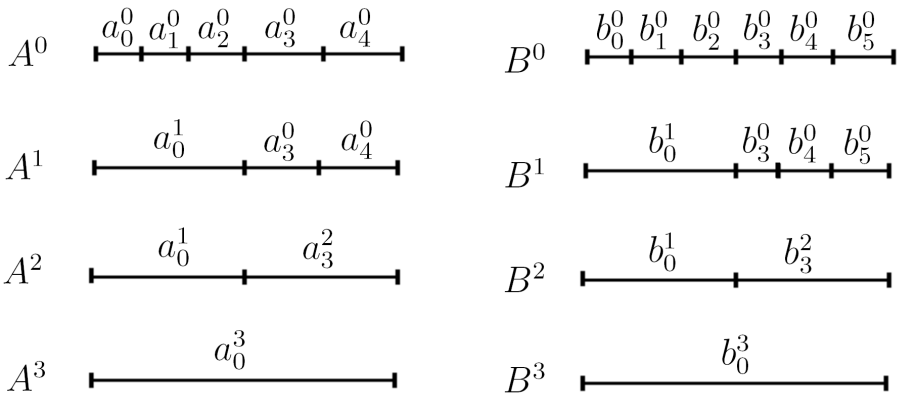


FIG. 5.11 – Exemple de chaînes avec les réécritures correspondantes.

$$\begin{cases} A^0 = a_0^0, a_1^0, a_2^0, a_3^0, a_4^0 & \\ A^1 = a_0^1, a_3^0, a_4^0 & (a_0^1 \rightarrow a_0^0, a_1^0, a_2^0) \\ A^2 = a_0^1, a_3^2 & (a_3^2 \rightarrow a_3^0, a_4^0) \\ A^3 = a_0^3 & (a_0^3 \rightarrow a_0^1, a_3^2) \end{cases} \quad \begin{cases} B^0 = b_0^0, b_1^0, b_2^0, b_3^0, b_4^0, b_5^0 & \\ B^1 = b_0^1, b_3^0, b_4^0, b_5^0 & (b_0^1 \rightarrow b_0^0, b_1^0, b_2^0) \\ B^2 = b_0^1, b_3^2 & (b_3^2 \rightarrow b_3^0, b_4^0, b_5^0) \\ B^3 = b_0^3 & (b_0^3 \rightarrow b_0^1, b_3^2) \end{cases}$$

L'application direct de l'algorithme de Dijkstra sur le graphe présenté sur la figure 5.12 fournit un appariement équivalent à celui calculé dans la section 5.2.3.

5.2.5 Exemple d'utilisation

Nous allons illustrer le fonctionnement de notre algorithme sur un exemple simple consistant à mettre en correspondance les deux séquences de brins $a_i = \{a_0, \ldots, a_{11}\}$ et $b_i = \{b_0, \ldots, b_{11}\}$ correspondant à la figure. 5.13 représentant deux triangles à différents niveaux de finesses.

Pour illustrer cet exemple nous utiliserons des coups fixes pour chacunes des opérations élementaires :

- un coût de suppression $CoutSuppression = 10$,
- un coût de transformation (via réécriture) $CoutDeTransformation = 0.4$ et
- un coût d'appariement (diagonal) $CoutDiag$ à 1.

Le tableau de coûts $\Delta[i, j]$ correspondant, illustré sur la figure. 5.15, est construit dynamiquement en utilisant l'équation 5.16 et l'algorithme 5. La matrice de coût correspondant à l'appariement de ces deux chaînes sans utiliser de réécritures est présenté sur la figure 5.14. On peut déjà noter que le fait d'utiliser les réécritures diminue le coût global d'appariement.

L'appariement débute par l'appariement de $a_0 \leftrightarrow b_0$. Le coût de 11 dans la case $M_{0,1}$ correspond au coût d'appariement de $a_0 a_1 \leftrightarrow b_0$, soit 1 pour $a_0 \leftrightarrow b_0$ plus un coût de suppression de 10. Le coût de 1,4 dans la case $M_{0,4}$ correspond au coût d'appariement de la réécriture de $a_0 a_1 a_2 a_3$ en a_0 avec b_0, soit 1 pour la mise en correspondance plus un coût

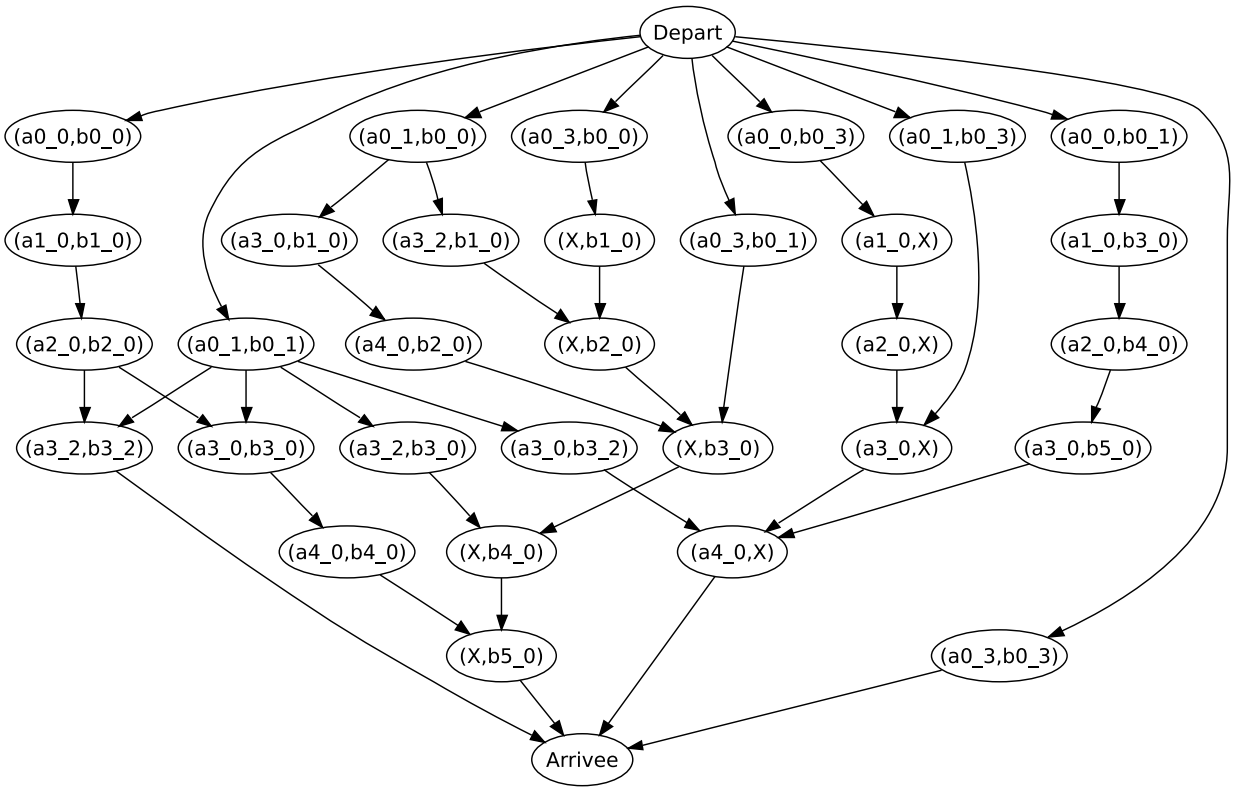


FIG. 5.12 – Exemple de graphe acyclique correspondant à la mise en correspondance des deux chaîne présentée à la figure 5.11. Le sommet de la fomre (a_i, X) codent la suppression de l'élément a_i, de même que les sommets de la forme (X, b_j) codent la suppresion de l'élément b_j.

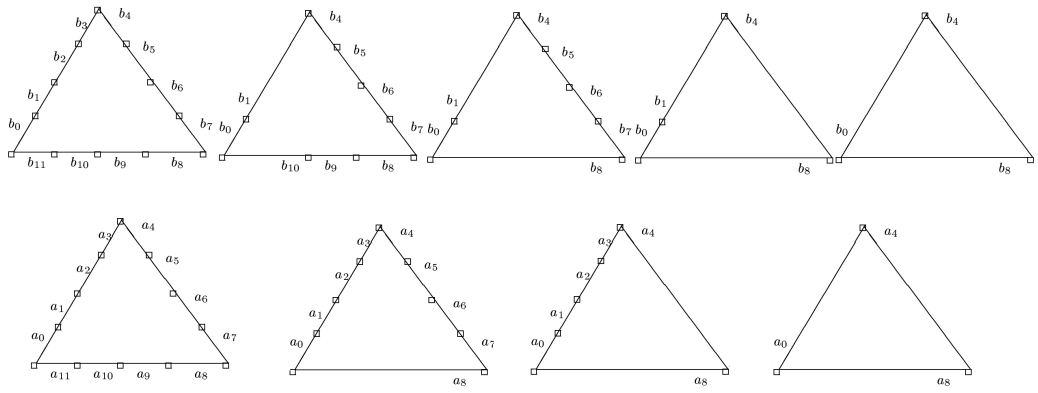


FIG. 5.13 – Exemple de réécritures possibles d'une même région représentée à différents niveaux dans deux pyramides.

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}
b_0	1	11	21	31	41	51	61	71	81	91	101	111
b_1	11	2	12	22	32	42	52	62	72	82	92	102
b_2	21	31	3	13	23	33	43	53	63	73	83	93
b_3	31	41	51	4	14	24	34	44	54	64	74	84
b_4	41	51	61	71	5	15	25	35	45	55	65	75
b_5	51	61	71	81	91	6	16	26	36	46	56	66
b_6	61	71	81	91	101	111	7	17	27	37	47	57
b_7	71	81	91	101	111	121	131	8	18	28	38	48
b_8	91	101	111	121	131	141	151	161	9	19	29	39
b_9	101	111	121	131	141	151	161	171	181	10	20	30
b_{10}	111	121	131	141	151	161	171	181	191	1	11	21
b_{11}	121	131	141	151	161	171	181	191	201	211	2	12

FIG. 5.14 – Tableau $R[i,j]$ représentant les coûts minimaux d'appariement des séquences de brins relatives à l'exemple de la fig. 5.13 sans prendre en compte les réécritures.

fixe de transformation de 0.4. L'ensemble de la matrice est ainsi remplie, en sauvegardant en parallèle les différentes opérations effectuées dans la matrice de $retour$.

L'appariement final pour cet exemple est donc retrouvé grâce au tableau $retour$ illustré sur la figure 5.16. En partant de la dernière case $M_{11,11} = (a_{11}, b_{11})$ le 3 indique une réécriture du brin b_8, b_9, b_{10}, b_{11} en b_8. Cette réécriture nous amène sur la case (a_{11}, b_8) dans laquelle le -3 code une réécriture du brin a_8, a_9, a_{10}, a_{11} en a_8. Nous arrivons alors sur la case (a_8, b_8) dans laquelle le 0.5 rencontré nous amène à apparier les brins a_8 et b_8 .

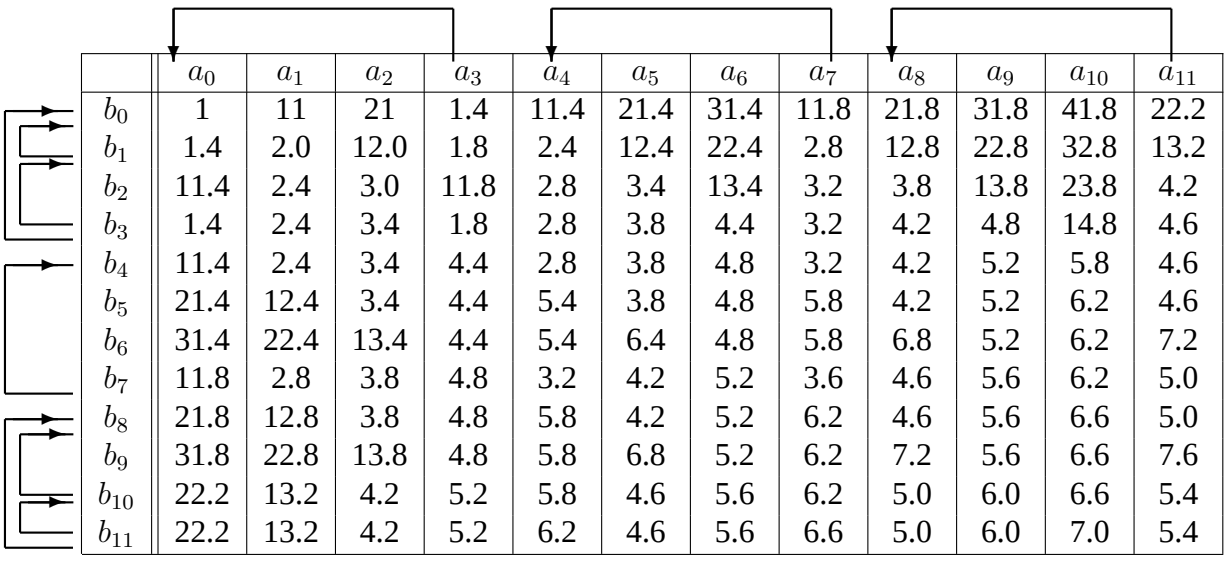
En remontant la matrice jusqu'à arriver sur la case (a_0, b_0),l'appariement optimal fournit par l'algorithme met finalement en correspondance les couples de (brin,niveau) suivants : $\{((a_0, 3) \leftrightarrow (b_0, 4)), ((a_4, 2) \leftrightarrow (b_4, 3)), ((a_8, 1) \leftrightarrow (b_8, 2))\}$.

5.3 Définition des attributs et expérimentations

5.3.1 Définition des attributs

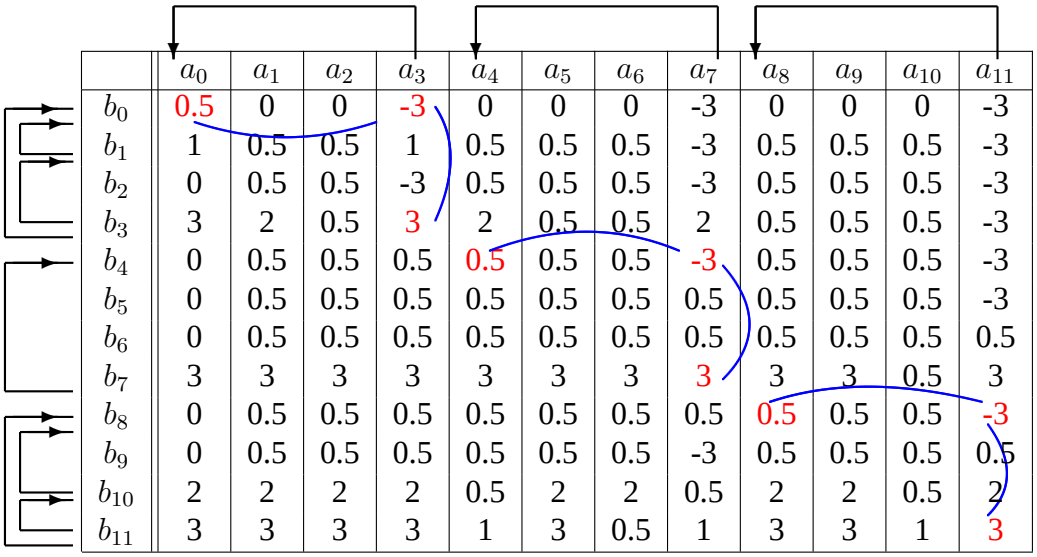
Étant donnés une pyramide P et un sommet $v \in \mathcal{V}_P$, nous noterons R_v la région géométrique associée au sommet v et ∂R_v la frontière de la région R_v. Le vecteur de caractéristiques, défini dans la section 5.1, sera composé dans les expériences qui vont suivre de $M = 6$ caractéristiques :

1. la première caractéristique $Feat_1(v)$ correspond à la taille en pixel de la région R_v. Nous pouvons noter, comme nous l'avions imposé dans la section 5.1, que cette caractéristique définit bien un critère croissant.
2. Les 3 caractéristiques $Feat_2(v)$, $Feat_3(v)$ et $Feat_4(v)$ correspondent à la moyenne de la région R_v pour chacunes des 3 composantes R,G et B.



	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}
b_0	1	11	21	1.4	11.4	21.4	31.4	11.8	21.8	31.8	41.8	22.2
b_1	1.4	2.0	12.0	1.8	2.4	12.4	22.4	2.8	12.8	22.8	32.8	13.2
b_2	11.4	2.4	3.0	11.8	2.8	3.4	13.4	3.2	3.8	13.8	23.8	4.2
b_3	1.4	2.4	3.4	1.8	2.8	3.8	4.4	3.2	4.2	4.8	14.8	4.6
b_4	11.4	2.4	3.4	4.4	2.8	3.8	4.8	3.2	4.2	5.2	5.8	4.6
b_5	21.4	12.4	3.4	4.4	5.4	3.8	4.8	5.8	4.2	5.2	6.2	4.6
b_6	31.4	22.4	13.4	4.4	5.4	6.4	4.8	5.8	6.8	5.2	6.2	7.2
b_7	11.8	2.8	3.8	4.8	3.2	4.2	5.2	3.6	4.6	5.6	6.2	5.0
b_8	21.8	12.8	3.8	4.8	5.8	4.2	5.2	6.2	4.6	5.6	6.6	5.0
b_9	31.8	22.8	13.8	4.8	5.8	6.8	5.2	6.2	7.2	5.6	6.6	7.6
b_{10}	22.2	13.2	4.2	5.2	5.8	4.6	5.6	6.2	5.0	6.0	6.6	5.4
b_{11}	22.2	13.2	4.2	5.2	6.2	4.6	5.6	6.6	5.0	6.0	7.0	5.4

FIG. 5.15 – Tableau $R[i,j]$ représentant les coûts minimaux d'appariement des séquences de brins relatives à l'exemple de la fig. 5.13. Les flèches traduisent les multiples réécritures possibles des différents brins



	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}
b_0	0.5	0	0	-3	0	0	0	-3	0	0	0	-3
b_1	1	0.5	0.5	1	0.5	0.5	0.5	-3	0.5	0.5	0.5	-3
b_2	0	0.5	0.5	-3	0.5	0.5	0.5	-3	0.5	0.5	0.5	-3
b_3	3	2	0.5	3	2	0.5	0.5	2	0.5	0.5	0.5	-3
b_4	0	0.5	0.5	0.5	0.5	0.5	0.5	-3	0.5	0.5	0.5	-3
b_5	0	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	-3
b_6	0	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
b_7	3	3	3	3	3	3	3	3	3	3	0.5	3
b_8	0	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	-3
b_9	0	0.5	0.5	0.5	0.5	0.5	0.5	-3	0.5	0.5	0.5	0.5
b_{10}	2	2	2	2	0.5	2	2	0.5	2	2	0.5	2
b_{11}	3	3	3	3	1	3	0.5	1	3	3	1	3

FIG. 5.16 – Exemple de tableau $Retour[i,j]$, représentant les différentes opérations à effectuer pour retrouver l'appariement optimal relatif à l'exemple de la Fig. 5.13 et aux coûts calculés dans le tableau de la Fig. 5.15.

3. $Feat_5(v)$ correspond à la taille en lignels de la frontière ∂R_v et
4. $Feat_6(v)$ prend la valeur moyenne du gradient, calculée le long de la frontière ∂R_v.

Lors de l'étape d'appariement circulaire de chaînes (section 5.2), pour un brin d défini à un niveau i de la pyramide, nous caractérisons la forme de la frontière orientée associée au brin d par l'intermédiaire d'une fonction qui va associer à chacun des points de la frontière, la courbure [MALGOU08] de la frontière en ce point. Nous obtenons ainsi un vecteur de courbures que nous compressons en un vecteur de 8 caractéristiques en utilisant les huit premiers moments de Legendre, calculés par le biais d'une méthode rapide proposée par Shen [SHEN96].

Les polynômes de Legendre normalisés sur un intervalle $[-k, k]$ sont définis par :

$$L_n(x) = \frac{d^n}{dx^n}\frac{(x^2-k^2)^n}{(2k)^n.n!} \tag{5.17}$$

Les moments de legendre en zéros d'un signal S correspondent au produit scalaire de S avec les différents polynômes. Chaque moment M_n est donc défini par :

$$M_n =< S, L_n >= \int_{-k}^{k} S(t).L_n(t)dt$$

L'objet de l'optimisation introduite par Shen, consiste à éviter le calcul explicite des différents polynômes de Legendre. Shen montre pour cela que les moments de Legendre vérifient :

$$\forall i \geq 0 \begin{cases} < L_0, S_i > &= S_{i+1}(k) - S_{i+1}(-k) \\ < L_1, S_i > &= S_{i+1}(k) + S_{i+1}(-k) - \frac{1}{k} < L_0, S_{i+1} > \\ < L_n, S_i > &= < L_{n-2}, S_i > -\frac{2n-1}{k} < L_{n-1}, S_{i+1} >, \quad \forall n \geq 2 \end{cases}$$

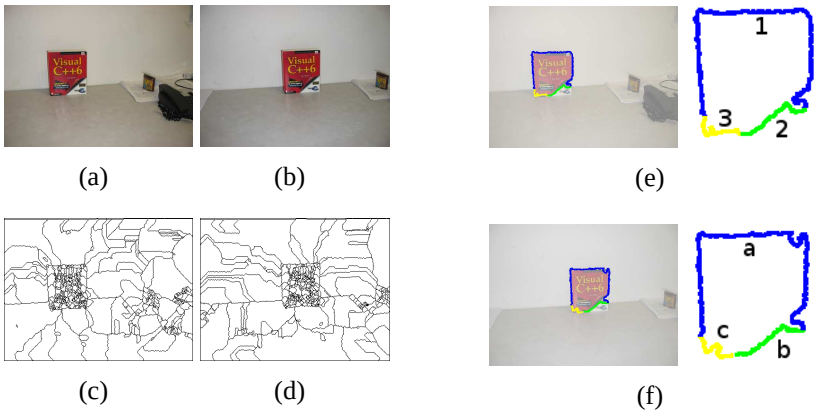
où les S_i correspondent aux moment cumulés d'ordre i du signal S. On posera par convention $S_0 = S$.

En partant des polynômes de Legendre d'ordre 0 et 1 défini par :

$$\begin{cases} L_0(x) &= 1 \\ L_1(x) &= \frac{d}{dx}\frac{(x^2-k^2)}{(2k)} = \frac{x}{k} \end{cases}$$

des calculs élémentaires détaillés en annexe (section A.3) nous fournissent les 8 premiers moments de Legendre :

1. $M_0 =< L_0, S >$
2. $M_1 =< L_1, S >$
3. $M_2 =< L_0, S > -\frac{3}{k} < L_1, S_1 >$
4. $M_3 =< L_1, S > -\frac{5}{k} < L_0, S_1 > +\frac{15}{k^2} < L_1, S_2 >$
5. $M_4 =< L_0, S > -\frac{10}{k} < L_1, S_1 > +\frac{35}{k^2} < L_0, S_2 > -\frac{105}{k^3} < L_1, S_3 >$
6. $M_5 =< L_1, S > -\frac{14}{k} < L_0, S_1 > +\frac{105}{k^2} < L_1, S_2 > -\frac{315}{k^3} < L_0, S_3 > +\frac{945}{k^4} < L_1, S_4 >$
7. $M_6 =< L_0, S > -\frac{10}{k} < L_1, S_1 > +\frac{35}{k^2} < L_0, S_2 > -\frac{105}{k^3} < L_1, S_3 >$



(a) (b) (e)

(c) (d) (f)

FIG. 5.17 – Deux frontières mises en correspondance (e) et (f) calculées à partir de deux pyramides obtenues à partir des partitions initiales (c) et (d) issues des images originales (a) et (b).

8. $M_7 =< L_1, S > + \frac{1}{k} < L_0, S_1 > + \frac{168}{k^2} < L_1, S_2 > + \frac{2142}{k^3} < L_0, S_3 > - \frac{15435}{k^4} < L_1, S_4 > + \frac{45045}{k^5} < L_0, S_5 > - \frac{135135}{k^6} < L_1, S_6 >$
9. $M_8 =< L_0, S > + \frac{6}{k} < L_1, S_1 > - \frac{216}{k^2} < L_0, S_2 > - \frac{4410}{k^3} < L_1, S_3 > - \frac{38115}{k^4} < L_0, S_4 > - \frac{249480}{k^5} < L_1, S_5 > - \frac{675675}{k^6} < L_0, S_6 > - \frac{2027025}{k^7} < L_1, S_7 >$

L'intérêt principal lié à cette méthode tient au fait que les expressions correspondant à chacun des moments peuvent être calculées en amont. La complexité liée à l'utilisation de ces moments est donc réduite au calcul des moments cumulés S_i et à l'évaluation des expressions ci dessus, ce qui permet d'accélérer de façon conséquente l'ensemble des calculs.

Nous ajoutons à ces 8 caractéristiques géométriques, trois caractéristiques colorimétriques correspondant à la couleur moyenne du sommet $\sigma_i^*(\alpha_i(d))$ (voir Section 5.2). La fonction de distance δ entre deux brins est alors définie comme la norme infinie d'un vecteur f en utilisant une étape de normalisation équivalente à celle utilisée dans l'équation 5.6 mais appliquée sur les caractéristiques des deux brins.

5.3.2 Résultats expérimentaux

Nous proposons dans cette section d'évaluer les performances de notre algorithme d'un point de vue qualitatif et quantitatif. D'un point de vue qualitatif nous présentons des résultats visuels de mises en correspondance de frontières sur différents types d'image. D'un point de vue quantitatif nous proposons une étude de l'importance relative des critères utilisés lors de l'étape de préfiltrage ainsi que l'influence de la finesse des segmentations initiales sur les temps d'éxecution des différentes étapes de notre algorithme.

Nous présentons sur la figure 5.17 un premier résultat de notre algorithme exécuté sur une paire d'images naturelles, représentant une même scène, prise avec un point de vue différent. Les cartes combinatoires codant les niveaux de base des deux pyramides $P = (G_0, \ldots, G_n)$ et $P' = (G'_0, \ldots, G'_{n'})$ sont obtenues par le biais d'un algorithme de ligne de partage des eaux [BRUN05B] appliqué à chacune des images d'entrée. Le paramètre α du gradient de Deriche réglant la finesse de la segmentation a été, dans ce cas, fixé à 0,5. Le schéma de construction permettant d'obtenir les niveaux suivants dans chacune

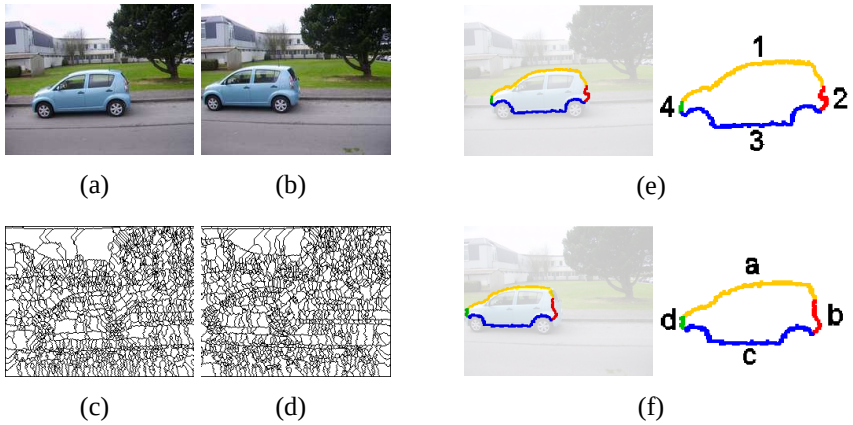


FIG. 5.18 – Deux frontières mises en correspondance (e) et (f) calculées à partir de deux pyramides obtenues à partir des partitions initiales (c) et (d) issues des images originales (a) et (b).

des pyramides utilise les heuristiques de fusion décrites dans la chapitre 3 de ce mémoire (voir également [PRUVOT07]). Les deux sommets v et v' appariés représentent la partie supérieure gauche du livre dans les deux images. En utilisant nos règles de réécriture, les deux séquences B_v et $B_{v'}$ représentant le contour du livre dans les cartes combinatoires des niveaux de base (Fig. 5.17 (c) et (d)) ont été regroupées en seulement 3 brins (Fig. 5.17 (e) et (f)) :

- un brin séparant le livre du mur situé derrière ($1 \leftrightarrow a$) ;
- un autre pour séparer la partie rouge du livre de la partie blanche ;
- un dernier permettant de séparer le livre de la table sur lequel il repose.

Tous ces brins sont appariés à des niveaux différents dans les pyramides P et P'. Par exemple le brin 1 appartenant au 838^{ime} niveau de P est mis en correspondance avec le brin a du niveau 961 dans la pyramide P'.

Nous présentons sur la figure 5.18 les résultats obtenus sur une paire d'images plus complexes. Le paramètre α réglant la finesse de la segmentation a été cette fois ci réglé à 0.8. Dans cet exemple les deux sommets v et v' appariés correspondent à la partie supérieure de la voiture dans les deux images. Les deux séquences B_v et $B_{v'}$ permettant de coder le contour de la voiture dans les cartes des niveaux de base pour chacune des pyramides P et P' (Fig. 5.18 (c) et (d)) ont été regroupées en 4 brins (Fig. 5.18 (e) et (f)). Les brins appariés appartiennent encore une fois à des niveaux différents dans chacune des pyramides P et P', comptant respectivement 1839 et 1953 niveaux au total. Le résultat du processus d'appariement hiérarchique pour cet exemple est l'ensemble des couples de $(brin, niveau)$ suivant :

$$\begin{cases} (a, 1821) \leftrightarrow (1, 1929) \\ (b, 1797) \leftrightarrow (2, 1908) \\ (c, 1779) \leftrightarrow (3, 1902) \\ (d, 1779) \leftrightarrow (4, 1899) \end{cases}$$

Nous présentons de la même façon sur la figure 5.19 différents résultats obtenus pour d'autres images (ourson, grille-pain, pomme, container) dont certaines présentent un arrière plan plus complexe. Les résultats sont similaires aux résultats obtenus pour les images

précédentes et nous fournissent des ensembles de couples de brins appariés à différents niveaux. Tous ces résultats ont été obtenus avec le paramètre $\alpha = 0.5$.

Pour mesurer l'importance des différentes caractéristiques utilisées lors de l'étape de pré-filtrage, nous avons observé pour trois images (voiture, livre et container) le positionnement du couple de régions sélectionné pour l'appariement dans les exemples précédents. Nous avons ainsi mesuré la modification du rang du couple de régions selectionné, induite par le retrait d'une caractéristique. Plus les écarts de positionnement sont importants et plus l'attribut correspondant peut être considéré comme important pour cette étape. L'analyse du tableau 5.2 présentant ces écarts de positions, tend à prouver que le gradient moyen le long du contour est une caractéristique essentielle. Cette caractéristique qui est la seule qui prennent en compte directement le voisinage de la région, occasionne un écart de position important pour les trois images testées lorsqu'elle n'est pas prise en compte. La première caractéristique qui permet de comparer la taille en pixels des régions se révèle également une caractéristique primordiale alors que la taille en lignels de la frontière semble moins importante au bon déroulement de cette étape de préfiltrage.

caractéristique rétirée	voiture	livre	container
taille en pixels	+26	>100	+38
moyenne des 3 composantes RVB	+18	+14	+26
taille de la frontière en lignels	+11	+11	+16
gradient moyen le long de la frontière	>100	>100	+87

TAB. 5.2 – Écarts relatifs de positionnement du couple de régions sélectionné pour l'appariement lors de l'étape de préfiltrage circulaire en enlevant une à une les différentes caractéristiques. Ces écarts ont été mesurés pour les 3 images (voiture, livre et container).

L'ensemble des résultats présentés dans cette section est basé sur des hiérarchies de coupes comme nous les avons définies dans le chapitre 3 . Nous avons effectué les mêmes opérations à partir de hiérarchies binaires et si nous sélectionnons les mêmes couples de régions au départ, nous obtenons exactement les mêmes résultats, ce qui n'a rien d'étonnant étant donné que les hiérarchies de coupes forment un sous ensemble des hiérarchies binaires. Le tableau 5.3 présente les temps d'exécution pour ces deux types de hiérarchies. Nous pouvons observer que même si le temps nécessaire à la construction d'une hiérarchie de coupes et légèrement plus important que pour une hiérarchie binaire, les étapes de préfiltrage et d'appariement circulaire sont largement plus rapide avec les hiérarchies de coupes composées de moins de niveaux.

L'utilisation de hiérarchies de coupes nous permet d'obtenir les mêmes résultats deux à quatre fois plus rapidement qu'avec des hiérarchies binaires. Nous pouvons également noter que pour l'image de la voiture nous avons du interrompre le processus d'appariement après plus de deux heures de traitement, dans le cas de hiérarchies binaires, sans obtenir aucun résultat.

Nous avons mesuré le comportement de notre algorithme en fonction du nombre de régions dans les partitions initiales, en faisant varier le paramètre α du gradient de Deriche entre 0.3 et 1. En sélectionnant le même couple de régions à apparier, nous obtenons des résultats similaires mais les temps d'exécution de l'étape de préfiltrage aussi bien que ceux nécessaires à l'étape d'appariement augmentent lorsque le nombre de régions dans les

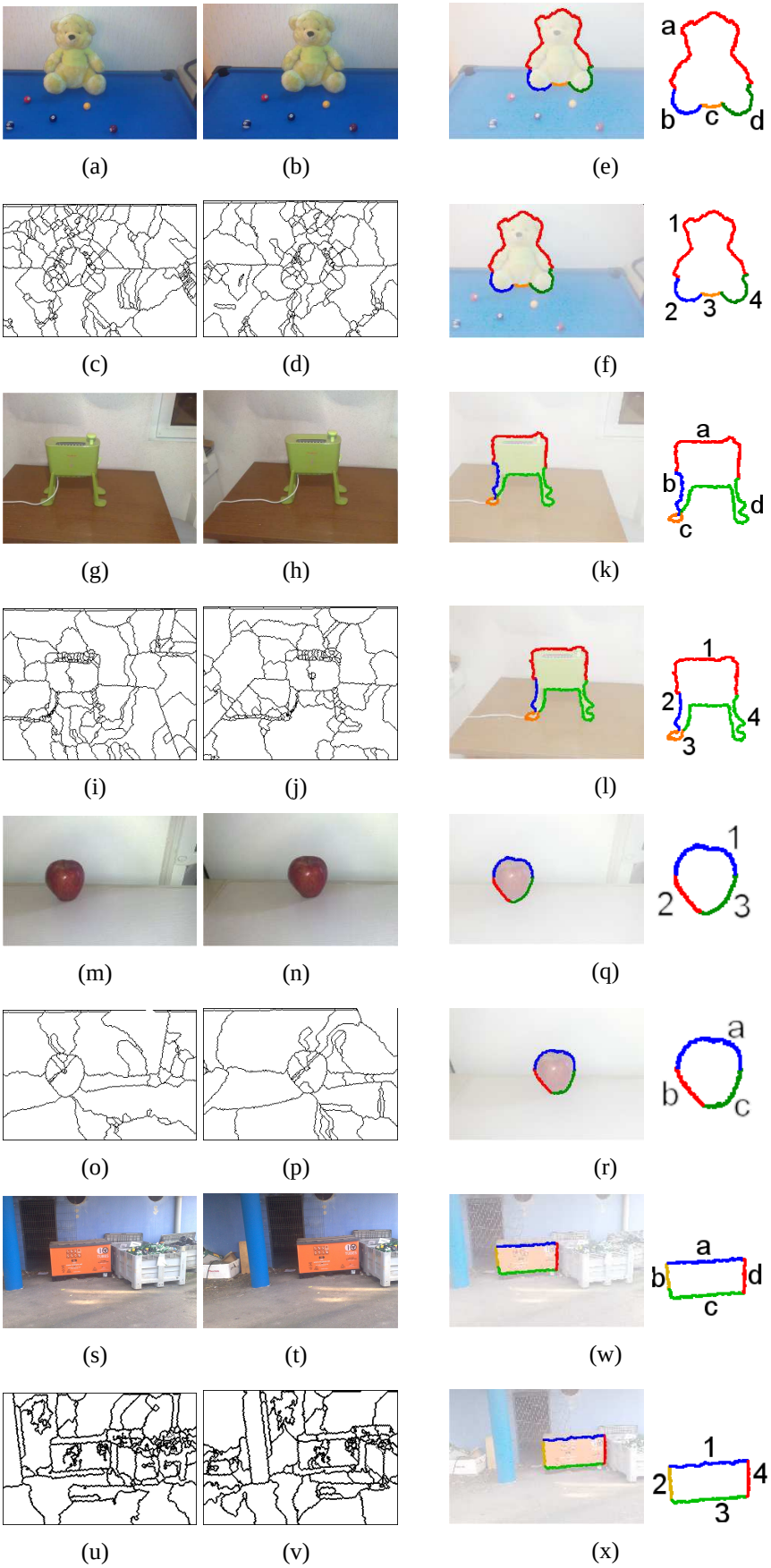


FIG. 5.19 – Mises en correspondance de frontières, calculées à partir de pyramides.

	avec hiérarchie de coupes				avec hiérarchie binaire			
image	construction	préfiltrage	app.	Total	construction	préfil.	app.	Total
livre	2’08	12’32	2’06	16’46	1’16	46’16	3’06	50’50
voiture	2’31	28’41	4’37	35’49	1’37	>2h	-	-
grille-pain	1’32	11’12	2’28	15’12	0’57	47’32	2’56	51’25
ourson	1’17	14’27	1’48	17’32	0’46	62’51	2’24	66’01
pomme	0’41	6’11	0’56	7’50	0’28	9’37	0’59	11’04
container	1’36	13’37	2’11	17’24	1’07	37’51	2’54	41’52

TAB. 5.3 – Temps d’exécution correspondant à chacune des étapes de l’algorithme de mise en correspondance. Les temps sont donnés pour deux types de hiérarchies : des hiérarchies de coupes ainsi que des hiérarchies binaires classiques. Pour chacun de ces deux cas, sont donnés les temps nécessaires à la construction de la hiérarchie, à l’étape de pré-filtrage ainsi que le processus d’appariement circulaire (colonne «app.»).

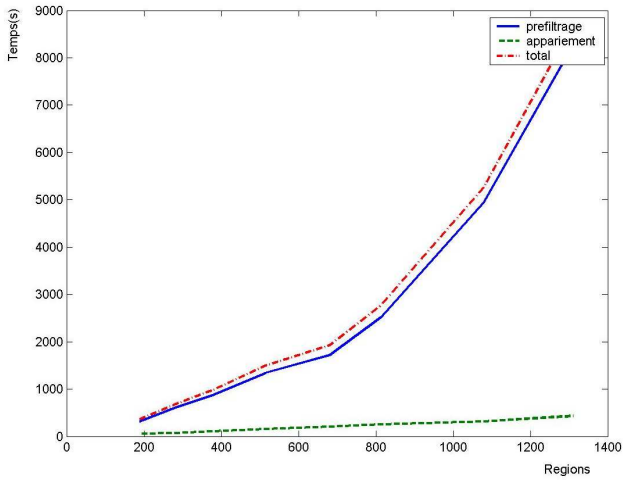


FIG. 5.20 – Temps nécessaire pour effectuer l’étape de préfiltrage (— bleu) l’étape d’appariement circulaire (- - - vert) ainsi que le temps total (.-.-. rouge) par rapport au nombre moyen de régions dans les deux images à apparier pour l’image de l’ourson.

partitions initiales augmente (voir Fig. 5.20). Notre algorithme reste néanmoins robuste pour un nombre de régions important, mais l’utilisation de partitions très fines ne se révèle pas forcement nécessaire pour obtenir un résultat satisfaisant, à partir du moment ou l’objet d’intérêt est correctement détouré par la partition initiale et où il présente un contraste suffisant avec le fond pour apparaître à un certain niveau de la pyramide.

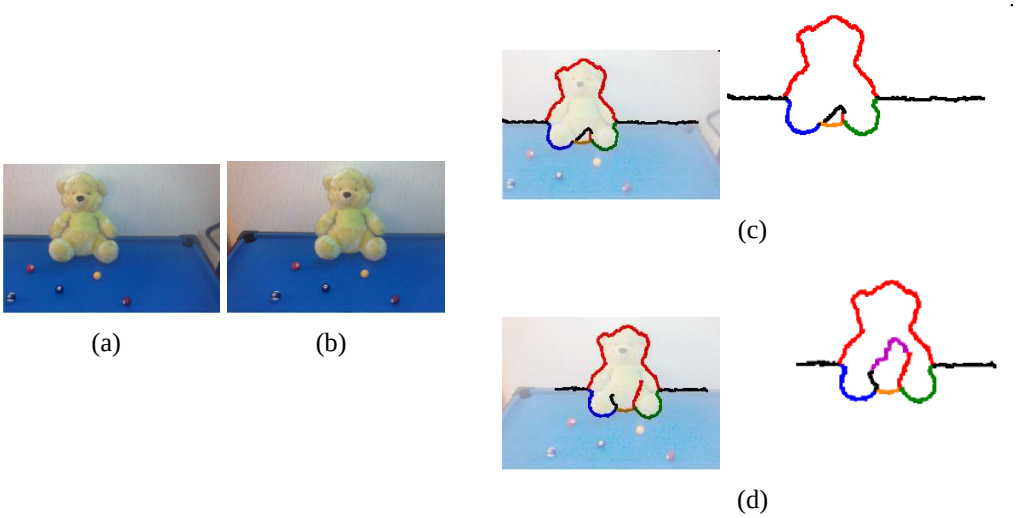
CONCLUSION ET PERSPECTIVES

NOUS avons présenté dans le chapitre 2 un état de l'art des différentes structures pyramidales permettant de représenter des hiérarchies de partitions.
Munis d'une vue d'ensemble des différentes représentations pyramidales, nous avons présenté, dans le chapitre 3, différentes heuristiques permettant la construction de hiérarchies de partitions. Les méthodes de construction que nous avons proposées, permettent d'obtenir des segmentations stables et robustes tout en proposant un compromis entre le temps d'exécution et l'énergie des partitions obtenues. Cette description est complétée par une étude expérimentale aussi bien qualitative que quantitative des différentes approches proposées, démontrant que les courbes d'énergies obtenues avec l'un de nos algorithmes sont proches de la courbe d'énergie minimale en dessous de laquelle aucun algorithme de segmentation ne peut descendre. Ce résultat étant loin d'être évident, a priori, puisque cette heuristique n'explore qu'une sous-partie de l'espace de recherche.

Nous avons ensuite présenté dans le chapitre 4 un état de l'art portant sur les principales méthodes d'appariement de graphes planaires : les approches algorithmiques puis les approches par optimisation. En considérant les principales approches d'appariement structurel nous avons proposé, dans le chapitre 5, une nouvelle approche au problème d'appariement de régions.
Nous avons proposé de combiner plusieurs approches d'appariement en tirant parti des principaux avantages liés à ces méthodes. Dans un premier temps nous construisons deux hiérarchies de partitions dans le but d'effectuer un appariement hiérarchique comme le propose Glantz [GLANTZ03]. La construction de ces hiérarchies se base sur les critères énergétiques définis dans le chapitre 3 et utilise les heuristiques de fusion que nous avons proposées dans les sections 3.2 et 3.3. Nous obtenons ainsi des segmentations robustes aux variations locales. Nous initialisons ensuite l'appariement en cherchant parmi les deux hiérarchies, un couple de régions dont les caractéristiques maximisent un critère de ressemblance en prenant également en compte des informations issues des voisinages orientés des régions à mettre en correspondance. Cette dernière étape est réalisée en utilisant un algorithme calculant une distance d'édition sur les chaînes représentant les frontières des deux régions. Ces chaînes sont représentées à différents niveaux dans la pyramide grâce à des systèmes de réécriture.

En terme de travail d'implémentation :

Partant d'un codage des pyramides combinatoires définies par Luc Brun, nous avons implémenté un codage explicite de la hiérarchie, sous forme de dendrogramme. Les relations entre les deux structures sont réalisées au travers des labels de sommets et des étiquettes de brins. Ce codage explicite de la hiérarchie nous permet de calculer la séquence des coupes optimales définissant une pyramide combinatoire de taille réduite. Cette pyramide est définie à partir du dendrogramme et des coupes optimales en construisant l'en-



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

FIG. 6.1 – Deux frontières mises en correspondance avec en noir les brins correspondant aux brins suivants rencontrés en effectuant une rotation alpha pour chaque morceau de frontière au niveau pour lequel il a été sélectionné (c) et (d). Images originales (a) et (b).

semble des noyaux de contraction qui définissent ses différents niveaux. Le dendrogramme et la pyramide combinatoire des coupes optimales sont utilisés conjointement pour respectivement :

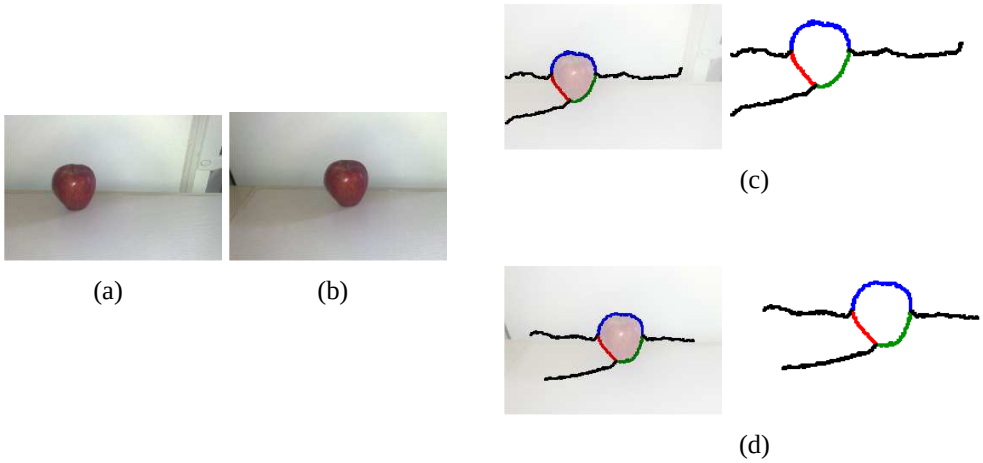
- préfiltrer notre base de régions ;
- effectuer l'appariement hiérarchique des contours des deux régions préfiltrées.

Notre travail personnel a été guidé par deux fils conducteur :

- proposer un méthode de segmentation hiérarchique permettant d'obtenir des partitions stables et robustes aux variations locales dans le but d'apparier les régions ;
- définir une nouvelle méthode d'appariement hiérarchique, permettant d'outrepasser les principaux inconvénients des méthodes existantes.

Les heuristiques de segmentation que nous avons définies sont basées sur le critère d'escalade proposé par Guigues et utilisent une approche *ascendante*. Ce type d'approche permet d'envisager un grand éventail de recherches. Les problématiques qui nous paraissent d'un intérêt majeur et qui demeurent ouvertes sont l'utilisation d'autre heuristiques de construction de hiérarchie.
Nous pourrions par exemple envisager des heuristiques *descendante* utilisant l'idée proposée dans la section 3.2 dans laquelle nous avons effectué des fusions successives par le biais d'un algorithme *union-find* fournissant une partition avec deux régions, permettant d'atteindre un minimum fixé a priori. Nous envisageons de renforcer cette heuristique en continuant le processus de découpage des régions, ce qui permettrait de fournir une partition moins sensible aux petites perturbations locales.

En ce qui concernent l'appariement de régions, de nombreux axes de recherche sont ouverts. Notre appariement se limite pour l'instant à une région dans chacunes des hiérarchies avec un voisinage appartenant à l'ensemble des coupes dans les pyramides. En utilisant l'orientation du plan, nous devrions pouvoir étendre cet appariement initial sans aboutir à une combinatoire trop importante.



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

FIG. 6.2 – Deux frontières mises en correspondance avec en noir les brins correspondant aux brins suivants rencontrés en effectuant une rotation alpha pour chaque morceau de frontière au niveau pour lequel il a été sélectionné (c) et (d). Images originales (a) et (b).

Les figures 6.1 et 6.2 présentent des mises en correspondance de frontières. Nous avons tracé pour chaque brin apparié b, la face $\sigma_i^*(\alpha_i(b))$ où i représente le niveau auquel b est apparié. On représente donc les faces de «l'autre côté» des brins appariés aux niveaux où ceux-ci sont définis. Nous pouvons observer que sur l'image de la pomme (Fig. 6.2) les nouvelles faces peuvent être mises en correspondance directement. Sur la figure de l'ourson (Fig. 6.1), les nouvelles faces de part et d'autre de l'ourson, codant respectivement le tapis du billard et le mur, peuvent également être mises en correspondance. Les faces internes de l'ourson semblent par contre être définies à deux niveaux différents dans les deux pyramides. De fait, la face représentée sur la figure 6.1(d), est nettement plus importante que celle représentée sur la figure 6.1(c). Toutefois, nous pouvons raisonnablement supposer l'existence d'une segmentation de la face interne de l'ourson de la figure 6.1(d) à un niveau inférieur de sa pyramide. Une des perspectives ouvertes par cette thèse devra rechercher parmi toutes les segmentations de cette face, une sous face pouvant être mise en correspondance avec la face interne de l'ourson définie sur la figure 6.1(c). Ce processus d'agrandissement de l'appariement devra être itéré jusqu'à ce que tous les élargissements possibles de l'appariement impliquent des faces trop différentes ou qu'une fonction de coût globale sur l'appariement dépasse un certain seuil.

Les cartes combinatoires offrant un codage naturel des relations d'inclusion, leur prise en compte lors du processus de mise en correspondance pourrait s'avérer cruciale pour guider certains choix et diminuer encore la complexité générale du problème.

Du point de vue plus théorique, une étude sur la notion de coupe dans une pyramide combinatoire pourrait nous fournir des informations intéressantes. Il pourrait notamment être intéressant d'étudier, sous quelles conditions il existe un isomorphisme entre une carte et une coupe issue d'une pyramide combinatoire ou encore d'approfondir l'étude des relations existant entre deux cartes combinatoires issues de deux coupes effectuées dans une même pyramide.

INDEX

BIBLIOGRAPHIE

[ABDULK98] A. M. ABDULKADER. « Parallel Algorithms for Labelled Graph Matching ». Thèse de doctorat, Colorado. School of Mines, 1998. PhD thesis.

[ABDULR98] M. ABDULRAHIM & M. MISRA. « A Graph Isomorphism Algorithm for Object Recognition ». *PAA*, 1(3), pages xx–yy, 1998.

[ALEXAN61] P. ALEXANDROFF. « Elementary Concepts of Topology ». Encyclopedia of Mathematics, New York, 1961.

[ARBELA05] P. ARBELAEZ & L. COHEN. « Segmentation d'Images Couleur par partitions de Voronoi ». *Traitement du Signal*, 21(5), pages 20–40, 2005.

[BASIN94] D. A. BASIN. « A Term Equality Problem Equivalent to Graph Isomorphism ». *Information Processing Letters*, 51(2), pages 61–66, 1994.

[BEN YA95] S. BEN YACOUB & J. M. JOLION. « Hierarchical Line Extraction ». *IEEE Vision, Image and Signal Processing*, 142(1), pages 7–14, 1995.

[BENZEC73] J. BENZECRI, rédacteur. « L'analyse des données. Tome I : La taxinomie. (French) ». Dunod. VIII, Paris - Bruxelles - Montreal, 1973.

[BERGE70] C. BERGE. « Graphes et Hypergraphes ». Dunod, Paris, 1970.

[BERGE74] C. BERGE & D. RAY-CHAUDHURI, rédacteurs. « Hypergraph Seminar ». Berlin, 1974. ISBN 3-540-06846-5.

[BERGE83] C. BERGE. « Graphes ». Gauthier-villars, bordas édition, 1983.

[BERZTI73] A. BERZTISS. « A Backtrack Procedure for Isomorphism of Directed Graphs ». *JACM*, 20(3), pages 365–372, July 1973.

[BIEDL04] T. BIEDL, E. D. DEMAINE, C. A. DUNCAN, R. FLEISCHER & S. G. KOBOUROV. « Tight bounds on maximal and maximum matching ». *Discrete Mathematics*, 285(Issues 1-3), pages 7–15, August 2004.

[BIENEN87] E. BIENENSTOCK & C. VON DER MALSBURG. « A neural network for invariant pattern recognition ». *Europhysics Letters*, vol. 4, pages 121–126, 1987.

[BISTER90] M. BISTER, J. CORNELIS & A. ROSENFELD. « A critical view of pyramid segmentation algorithms ». *Pattern Recognit Letter.*, 11(9), pages 605–617, September 1990.

[BLAKE87] A. BLAKE & A. ZISSERMAN. « Visual Reconstruction ». MIT Press, 1987. ISBN 0-262-02271-0.

[BOMZE97] I. M. BOMZE. « Evolution towards the Maximum Clique ». *J. of Global Optimization*, 10(2), pages 143–164, 1997. ISSN 0925-5001.

[BOMZE00] I. M. BOMZE, M. PELILLO & V. STIX. « Approximating the Maximum Weight Clique Using Replicator Dynamics ». *IEEE-NN*, 11(6), page 1228, November 2000.

[BR96] L. BRUN. « Segmentation d'images couleur à base Topologique ». Thèse de doctorat, Université Bordeaux I, 351 cours de la Libération 33405 Talence, December 1996.

[BRUN99] L. BRUN & W. KROPATSCH. « Dual Contractions of Combinatorial Maps ». Rapport technique 54, Institute of Computer Aided Design, Vienna University of Technology, lstr. 3/1832,A-1040 Vienna AUSTRIA, January 1999. (contribution 70%).

[BR00A] L. BRUN & W. KROPATSCH. « The Construction of Pyramids with Combinatorial Maps ». Rapport technique 63, Institute of Computer Aided Design, Vienna University of Technology, lstr. 3/1832,A-1040 Vienna AUSTRIA, June 2000.

[BR00B] L. BRUN & M. MOKHTARI. « Two High Speed Color Quantization Algorithms ». Dans CÉPADUÈS, rédacteur, *Proceedings of CGIP'2000*, pages 116–121, Saint Etienne, October 2000. (contribution 50%).

[BRUN02] L. BRUN. « Traitement d'images couleur et pyramides combinatoires ». Habilitation à diriger des recherches, Université de Reims, 2002.

[BRUN03A] L. BRUN & W. KROPATSCH. « Combinatorial Pyramids ». Dans SUVISOFT, rédacteur, *IEEE International conference on Image Processing (ICIP)*, vol. II, pages 33–37, IEEE, Barcelona, September 2003.

[BRUN03B] L. BRUN & W. KROPATSCH. « Construction of Combinatorial Pyramids ». Dans E. HANCOCK & M. VENTO, rédacteurs, *Graph based Representations in Pattern Recognition*, vol. 2726, pages 1–12, IAPR-TC15, York, UK, June 2003.

[BRUN05A] L. BRUN & W. KROPATSCH. « Inside and Outside within Combinatorial Pyramids ». Dans L. BRUN & M. VENTO, rédacteurs, *5th IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition*, IAPR-TC15, Springer, Berlin Heidelberg, New York, Poitiers (France), April 2005. ISBN : 3-540-25270-3.

[BRUN05B] L. BRUN, M. MOKHTARI & F. MEYER. « Hierarchical watersheds within the Combinatorial Pyramid framework ». Dans *Proc. of DGCI 2005*, pages 34–44, IAPR-TC18, LNCS, 2005.

[BRUN06] L. BRUN & W. KROPATSCH. « Contains and Inside relationships within combinatorial Pyramids ». *Pattern Recognition*, 39(4), pages 515–526, April 2006.

[BUNKE83] H. BUNKE. « Inexact Graph Matching for Structural Pattern Recognition ». *Pattern Recognition Letters*, 1(4), pages 245–253, 1983.

[BUNKE93A] H. BUNKE. « Recent advances in string matching ». Dans H. BUNKE, rédacteur, *Advances in Structural and Syntactic Pattern Recognition*, pages 3–21. World Scientific, 1993.

[BUNKE93B] H. BUNKE & U. BUHLER. « Applications of approximate string matching to 2D shape recognition ». *Pattern Recognition*, 26(12), pages 1797–1812, December 1993.

[BUNKE95] H. BUNKE & B. MESSMER. « Efficient attributed graph matching and its application to image analysis ». Dans *CIAP*, pages 44–55, 1995.

[BURT81] P. BURT, T.-H. HONG & A. ROSENFELD. « Segmentation and Estimation of Image Region Properties Through Cooperative Hierarchial Computation ». *IEEE Transactions on Sustems, Man and Cybernetics*, 11(12), pages 802–809, December 1981.

[BURT83] P. BURT & E. ADELSON. « The laplacian Pyramid as a Compact Image Code ». *IEEE Transaction on Communications*, 31(4), pages 532–540, April 1983.

[CAIHUA95] W. CAIHUA & A. KEIICHI. « Region correspondence by inexact attributed planar graph matching ». Dans 5^{th} *International Conference on Computer Vision*, pages 440 – 447, June 1995.

[CHAN01] T. CHAN & L. VESE. « Active contours without edges ». *IEEE Trans. Image Processing*, 10(3), pages 266–277, February 2001.

[CHEN95] R.-J. CHEN & B.-C. CHIEU. « Three-dimensional morphological pyramid and its application to color image sequence coding ». *Signal Processing*, 44(2), pages 163–180, 1995.

[CHIARE96] E. CHIARELLO, J. M. JOLION & C. AMOROS. « Regions growing with the stochastic pyramid : application in landscape ecology ». *Pattern Recognition*, 29(1), pages 61–75, 1996.

[CHUNG97] F. R. K. CHUNG. « Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92) (Cbms Regional Conference Series in Mathematics) ». American Mathematical Society, February 1997. ISBN 0821803158.

[CINQU95] L. CINQUE, S. LEVIALDI & A. ROSENFELD. « Fast pyramidal algorithms for image thresholding ». *Pattern Recognition*, 28(6), pages 901–906, June 1995.

[CONTE06] D. CONTE, P. FOGGIA, J. JOLION & M. VENTO. « A graph-based, multi-resolution algorithm for tracking objects in presence of occlusions ». *Pattern Recognition*, 39(4), pages 562–572, April 2006.

[CORDEL04] L. CORDELLA, P. FOGGIA, C. SANSONE & M. VENTO. « A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs ». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, xx(yy), page zzz, October 2004.

[CORI75] R. CORI. « Un code pour les graphes planaires et ses applications ». Thèse de doctorat, Université Paris VII, 1975.

[CORNEI70] D. CORNEIL & C. GOTLIEB. « An Efficient Algorithm for Graph Isomorphism ». *JACM*, 17(1), pages 51–64, January 1970.

[CO63] R. T. COX. « The Algebra of Probable Inference ». *American Journal of Physics*, vol. 31, pages 66–67, jan 1963.

[DURAND99] P. J. DURAND, R. PASARI, J. W. BAKER, & C. CHE TSAI. « An Efficient Algo- rithm for Similarity Analysis of Molecules ». *Internet Journal of Chemistry*, vol. 2, 1999.

[EDMOND60] J. EDMONDS. « A combinatorial representation for polyhedral surfaces ». *Notices American Society*, vol. 7, 1960.

[ENGLER00] R. ENGLERT & W. G. KROPATSCH. « Image Structure From Monotonic Dual Graph Contraction ». Dans M. NAGL, A. SCHÜRR & M. MÜNCH, rédacteurs, *Applications of Graph Transformations with Industrial Relevance*, vol. Vol. 1799, pages 297–308, Springer, Berlin Heidelberg, New York, Kerkrade, Netherlands, 2000.

[FELZEN98] P. FELZENSZWALB & D. HUTTENLOCHER. « Image segmentation using local variation ». Dans *In Proceedings of IEEE Conference on CVPR, Santa Barbara, CA*, pages 98–104, June 1998.

[FOGGIA07] P. FOGGIA, G. PERCANNELLA, C. SANSONE & M. VENTO. « Assessing the Performance of a Graph-Based Clustering Algorithm ». Dans *Graph based Representation in Pattern Recognition*, pages 215–227, 2007.

[FOSSER03] P. FOSSER, R. GLANTZ, M. LOCATELLI & M. PELILLO. « Swap Strategies for Graph Matching ». Dans E. R. HANCOCK & M. VENTO, rédacteurs, *Graph Based Representations in Pattern Recognition, 4th IAPR International Workshop, GbRPR 2003, York, UK, June 30 - July 2, 2003, Proceedings*, vol. 2726, pages 142–153, Springer, 2003.

[GARETH78] A. GARETH & D. SINGERMAN. « Theory of maps on orientable surfaces ». *Proceedings of the London Mathematical Society*, 3(37), pages 273–307, 1978.

[GAREY90] M. R. GAREY & D. S. JOHNSON. « Computers and Intractability ; A Guide to the Theory of NP-Completeness ». W. H. Freeman & Co., New York, NY, USA, 1990. ISBN 0716710455.

[GDALYA99A] Y. GDALYAHU & D. WEINSHALL. « Flexible Syntactic Matching of Curves and Its Application to Automatic Hierarchical Classification of Silhouettes ». *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(12), pages 1312–1328, 1999. ISSN 0162-8828.

[GDALYA99B] Y. GDALYAHU & D. WEINSHALL. « Flexible Syntactic Matching of Curves and Its Application to Automatic Hierarchical Classification of Silhouettes ». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(12), pages 1312–1328, 1999.

[GEMAN84] S. GEMAN & D. GEMAN. « Stochastic Relaxation, Gibbs Distributions,and the Bayesian Restoration of Images ». *PAMI*, 6(6), pages 721–741, November 1984.

[GEMAN87] S. GEMAN & D. GEMAN. « Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images ». Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987. ISBN 0-934613-33-8.

[GEMAN90] S. GEMAN & D. GEMAN. « Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images ». Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990. ISBN 1-55860-125-2.

[GHAHRA80] D. GHAHRAMAN, A. WONG & T. AU. « Graph Optimal Monomorphism Algorithm ». *SMC*, vol. 10, pages 181–188, April 1980.

[GLANTZ03] R. GLANTZ, M. PELILLO & W. G. KROPATSCH. « Hierarchical Matching of Panoramic Images ». Dans *ICIAP '03*, page 328, IEEE Computer Society, 2003. ISBN 0-7695-1948-2.

[GOLD96] S. GOLD & A. RANGARAJAN. « A Graduated Assignment Algorithm for Graph Matching ». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4), pages 377–388, 1996.

[GORDON97] I. GORDON. « Theories of Visual Perception ». Johen Wiley and Son, 1997.

[GUIGUE03] L. GUIGUES. « Modèles Multi-Echelles pour la Segmentation d'Images ». Thèse de doctorat, Université de Cergy-Pontoise, 2003.

[GUIGUE06] L. GUIGUES, J. P. COCQUEREZ & H. MEN. « Scale-Sets Image Analysis ». *Int. J. Comput. Vision*, 68(3), pages 289–317, 2006. ISSN 0920-5691.

[HAXHIM02A] Y. HAXHIMUSA, R. GLANTZ, M. SAIB, G. LANGS & W. G. KROPATSCH. « Logarithmic Tapering Graph Pyramid ». Dans L. VAN GOOL, rédacteur, *Pattern Recognition, 24th DAGM Symposium*, vol. 2449, pages 117–124, Springer, Berlin Heidelberg, Zurich, Switzerland, September 2002.

[HAXHIM02B] Y. HAXHIMUSA, R. GLANTZ, M. SAIB, G. LANGS & W. G. KROPATSCH. « Reduction Factors of Pyramids on Undirected and Directed Graphs ». Dans H. WILDENAUER & W. G. KROPATSCH, rédacteurs, *Computer Vision - CVWW'02, Computer Vision Winter Workshop*, pages pp. 29–38, PRIP, TU Wien, Wien, Austria, February 2002.

[HAXHIM03] Y. HAXHIMUSA & W. KROPATSCH. « Hierarchical Image Partitioning with Dual Graph Contraction ». Dans B. MILAELIS & G. KRELL, rédacteurs, *Proceedings of DAGM Symposium*, vol. 2781, pages 338–345, Germany, Octobre 2003.

[HEBERT95] T. HEBERT & K. LU. « Expectation-maximization algorithms, null spaces, and MAP image restoration ». *IP*, 4(8), pages 1084–1095, August 1995.

[HOPCRO72] J. HOPCROFT & R. TARJAN. « Isomorphism of Planar Graphs ». Dans *CCComp*, pages 131–152, 1972.

[HUFFMA52] D. HUFFMAN. « A Method for the Construction of Minimum-Redundancy Codes ». *Proceedings of the IRE*, 40(9), pages 1098–1101, 1952.

[JOLI92] J. M. JOLION & A. MONTANVERT. « The Adaptative Pyramid : A framework for 2D image analysis ». *Computer Vision, Graphics, and Image Processing*, 55(3), pages 339–348, May 1992.

[JOLION93] J. M. JOLION. « A Hierarchical Framework for Robust Extraction and Delineation of Geometric Features ». *Pattern Recognition*, vol. 26, pages 1295–1304, 1993.

[JOLI94] J.-M. JOLION & A. ROSENFELD. « A Pyramid Framework for Early Vision ». Kluwer Academic Publishers, 1994.

[JOLI01] J.-M. JOLION. « Data Driven Decimation of Graphs ». Dans J.-M. JOLION, W. KROPATSCH & M. VENTO, rédacteurs, *Proceedings of* 3^{rd} *IAPR-TC15 Workshop on Graph based Representation in Pattern Recognition*, pages 105–114, Ischia-Italy, May 2001.

[KOEPFL94] G. KOEPFLER, C. LOPEZ & J. M. MOREL. « A Multiscale Algorithm for Image Segmentation by Variational Method ». *SIAM Journal on Numerical Analysis*, 31(1), pages 282–299, 1994.

[KONDOR02] R. I. KONDOR & J. LAFFERTY. « Diffusion kernels on graphs and other discrete input spaces ». Dans *Proceedings of the International Conference on Machine Learning (ICML)*, 2002.

[KONIK95] H. KONIK, A. TRÉMEAU & B. LAGET. « Multiresolution color image analysis ». Dans *Proceedings of the 3rd Color Imaging Conference*, pages 82–85, IS&T/SID, Scottsdale, nov 1995.

[KONIK96] H. KONIK, V. LOZANO & B. LAGET. « Color Pyramids For Image Processing ». *Journal of Imaging Science and Technology*, 40(6), pages 535–542, nov 1996.

[KOVALE89] V. A. KOVALEVSKY. « Finite topology as applied to image analysis ». *Comput. Vision Graph. Image Process.*, 46(2), pages 141–161, 1989. ISSN 0734-189X.

[KROPAT95A] W. G. KROPATSCH. « Building Irregular Pyramids by Dual Graph Contraction ». *IEE-Proc. Vision, Image and Signal Processing*, Vol. 142(No. 6), pages pp. 366–374, December 1995.

[KROPAT95B] W. G. KROPATSCH. « Equivalent Contraction Kernels and The Domain of Dual Irregular Pyramids ». Rapport technique PRIP-TR-42, Institute f. Automation 183/2, Dept. for Pattern Recognition and Image Processing, TU Wien, Austria, 1995.

[KROPAT95C] W. G. KROPATSCH & H. MACHO. « Finding the Structure of Connected Components Using Dual Irregular Pyramids ». Dans *Cinquième Colloque DGCI*, pages 147–158, LLAIC1, Université d'Auvergne, ISBN 2-87663-040-0, September 1995.

[KROPAT98] W. G. KROPATSCH. « From Equivalent Weighting Functions to Equivalent Contraction Kernels ». Dans E. WENGER & L. I. DIMITROV, rédacteurs, *Digital Image Processing and Computer Graphics (DIP-97) : Applications in Humanities and Natural Sciences*, vol. 3346, pages 310–320, SPIE, 1998.

[LECLER89] Y. G. LECLERC. « Constructing Simple Stable Descriptions for Image Partitioning ». *International Journal of Computer Vision*, 3(1), pages 73–102, 1989.

[LI01] S. Z. LI. « Markov random field modeling in image analysis ». Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001. ISBN 4-431-70309-8.

[LLADOS01] J. LLADOS, E. MARTI & J. VILLANUEVA. « Symbol Recognition by Error-Tolerant Subgraph Matching between Region Adjacency Graphs ». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10), pages 1137–1143, 2001. ISSN 0162-8828.

[LLADÓS97] J. LLADÓS. « Combining Graph Matching and Hough Transform for Hand-Drawn Graphical Document Analysis. Application to Architectural Drawings ». Phd dissertation, Universitat autonoma de Barcelona, November 1997.

[LOZANO98] V. LOZANO. « Contribution de l'analyse d'image couleur au traitement des images textile ». Thèse de doctorat, Université Jean Monnet, LIGIV, Saint-Etienne, France, 1998.

[LOZANO05] M. A. LOZANO & F. ESCOLANO. « Protein Classification with Kernelized Softassign ». Dans L. BRUN & M. VENTO, rédacteurs, *5th IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition*, vol. 3434, pages 331–340, IAPR-TC15, Springer, Berlin Heidelberg, New York, Poitiers(France), April 2005.

[M. LOC02] M. P. M. LOCATELLI. « Swaps, diversification, and the combinatorics of pivoting for the maximum weight clique ». Rapport technique, Rapporto di Ricerca CS-2002-12, Settembre 2002.

[MAES90] M. MAES. « On a cyclic string-to-string correction problem ». *Inf. Process. Lett.*, 35(2), pages 73–78, 1990. ISSN 0020-0190.

[MAES91] M. MAES. « Polygonal shape recognition using string-matching techniques ». *Pattern Recogn.*, 24(5), pages 433–440, 1991. ISSN 0031-3203.

[MALGOU08] R. MALGOUYRES, F. BRUNET & S. FOUREY. « Binomial Convolutions and Derivatives Estimation from Noisy Discretizations ». Dans D. COEURJOLLY, rédacteur, *Proceedings of DGCI'2008*, Springer Verlag, 2008.

[MALLAT96] S. MALLAT. « Wavelets for a vision ». *Proceedings of the IEEE*, 84(4), pages 604–614, 1996.

[MARFIL04] R. MARFIL, J. A. RODRIGUEZ, A. BANDERA & F. SANDOVAL. « Bounded irregular pyramid : a new structure for color image segmentation ». *Pattern Recognition*, 37(3), pages 623–626, March 2004. ISSN 0031-3203.

[MARFIL07] R. MARFIL, L. MOLINA-TANCO, A. BANDERA & F. SANDOVAL. « The Construction of Bounded Irregular Pyramids with a Union-Find Decimation Process ». Dans F. ESCOLANO & M. VENTO, rédacteurs, *Graph based Representation in Pattern Recognition'2007*, pages 307–318, IAPR TC15, LNCS, Alicante, June 2007.

[MASSAR02] A. MASSARO, M. PELILLO & I. M. BOMZE. « A Complementary Pivoting Approach to the Maximum Weight Clique Problem ». *SIAM J. on Optimization*, 12(4), pages 928–948, 2002. ISSN 1052-6234.

[MCGREG82] J. J. MCGREGOR. « Backtrack search algorithms and the maximal common subgraph problem ». Dans *Software - Practice and Experience*, vol. 12, pages 23–34, 1982.

[MEER88] P. MEER, S.-N. JIANG, E. S. BAUGHER & A. ROSENFELD. « Robustness of image pyramids under structural perturbations ». *Comput. Vision Graph. Image Process.*, 44(3), pages 307–331, 1988. ISSN 0734-189X.

[MEER89A] P. MEER. « Stochastic image pyramids ». *Computer Vision Graphics Image Processing*, vol. 45, pages 269–294, 1989.

[MEER89B] P. MEER & S. CONNELLY. « A Fast Parallel Method for Synthesis of Random Patterns ». *Pattern Recognition*, 22(2), pages 189–204, 1989.

[MESSME98] B. MESSMER & H. BUNKE. « A New Algorithm for Error Tolerant Subgraph Isomorphism Detection ». *PAMI*, 20(5), pages 493–504, May 1998.

[MOLLIN02] R. A. MOLLINEDA, E. VIDAL & F. CASACUBERTA. « A Windowed Weighted Approach for Approximate Cyclic String Matching ». *icpr*, vol. 04, page 40188, 2002. ISSN 1051-4651.

[MONTAN91] A. MONTANVERT, P. MEER & A. ROSENFELD. « Hierarchical Image Analysis Using Irregular Tessellations ». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4), pages 307–316, APRIL 1991.

[MOREL95] J. M. MOREL & S. SOLIMINI. « Variational methods in image segmentation ». Birkhauser Boston Inc., Cambridge, MA, USA, 1995. ISBN 0-8176-3720-6.

[MULHEM01] P. MULHEM, W. K. LEOW & Y. K. LEE. « Fuzzy Conceptual Graphs for Matching Images of Natural Scenes ». Dans *IJCAI*, pages 1397–1404, 2001.

[MUMFOR89] D. MUMFORD & J. SHAH. « Optimal Approximation by Piecewise Smooth Functions and Associated Variational Problems ». *Communications on Pure Applied Mathematics*, vol. 42, pages 577–685, 1989.

[MUNKRE57] J. MUNKRES. « Algorithms for the Assignment and Transportation Problems ». *Journal of the Society for Industrial and Applied Mathematics*, 5(1), pages 32–38, 1957.

[NEUHAU04] M. NEUHAUS & H. BUNKE. « An Error-Tolerant Approximate Matching Algorithm for Attributed Planar Graphs and Its Application to Fingerprint Classification ». Dans A. L. N. FRED, T. CAELLI, R. P. W. DUIN, A. C. CAMPILHO & D. DE RIDDER, rédacteurs, *SSPR/SPR*, vol. 3138, pages 180–189, Springer, 2004. ISBN 3-540-22570-6.

[OVERTU95] L. A. OVERTURF, M. L. COMER & E. J. DELP. « Color image coding using morphological pyramid decomposition ». *IEEE Transactions on Image Processing*, 4(2), pages 177–185, 1995.

[PALMER99] S. PALMER. « Vision Science :Photons to Phenomenology, Cambridge, MA : MIT Press ». Rapport technique, MIT, 1999.

[PAPKA99] R. PAPKA. « Online New Event Detection, Clustering, and Tracking ». Rapport technique UM-CS-1999-045, University of Massachusetts Amherst, 1999.

[PARDAL90] P. A. PARDALOS PM. « A global optimization approach for solving the maximum clique problem ». *International Journal of Computer Mathematics*, 11(33), pages 209–225, 1990.

[PELILL95] M. PELILLO & A. JAGOTA. « Feasible and infeasible maxima in a quadratic program for maximum clique ». *J. Artif. Neural Netw.*, 2(4), pages 411–420, 1995. ISSN 1073-5828.

[PELILL99] M. PELILLO, K. SIDDIQI & S. ZUCKER. « Continuous-based heuristics for graph and tree isomorphisms, with application to computer vision ». *Numerical Optimization Continuous and Discrete Problems*, 1999.

[PERIS02] G. PERIS & A. MARZAL. « Fast cyclic edit distance computation with weighted edit costs in classification ». *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 4, pages 184–187 vol.4, 2002. ISSN 1051-4651.

[POGGI85] T. POGGIO, V. TORRE & C. KOCH. « Computational vision and regularization theory ». *Nature*, vol. 317, pages 314–319, 1985.

[PRUVOT07] J. H. PRUVOT & L. BRUN. « Hierarchy Construction Schemes Within the Scale Set Framework ». Dans F. ESCOLANO & M. VENTO, rédacteurs, *Graph based Representation in Pattern Recognition'2007*, pages 126–137, IAPR TC15, LNCS, Alicante, June 2007.

[PYZLOT95] Z. PYZLOT, A. ROSENFELD & J. EPELBOIM. « An Exponential Pyramid Model of the Time-Course of Size Processing ». *Vision Res.*, 35(8), pages 1089–1107, 1995.

[RANGAR97] A. RANGARAJAN, H. CHUI & F. L. BOOKSTEIN. « The Softassign Procrustes Matching Algorithm ». Dans *IPMI*, pages 29–42, 1997.

[REYNER77] S. W. REYNER. « An analysis of a good algorithm for the subtree problem, correlated ». *SIAM J. Comput.*, 6(4), pages 730–732, 1977.

[RIESEN07] K. RIESEN, M. NEUHAUS & H. BUNKE. « Bipartite Graph Matching for Computing the Edit Distance of Graphs ». Dans F. ESCOLANO & M. VENTO, rédacteurs, *Graph based Representation in Pattern Recognition'2007*, pages 1–12, IAPR TC15, LNCS, Alicante, June 2007.

[RISSAN81] J. RISSANEN & G. G. L. JR. « Universal modeling and coding ». *IEEE Transactions on Information Theory*, 27(1), pages 12–22, 1981.

[ROBLES02] A. ROBLES-KELLY & E. R. HANCOCK. « String Edit Distance, Random Walks and Graph Matching ». Dans *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, pages 104–112, Springer-Verlag, London, UK, 2002. ISBN 3-540-44011-9.

[ROSENF98] A. ROSENFELD & C.-Y. SHER. « Detecting image primitives using feature pyramids ». *Journal of Information Sciences*, vol. 107, pages 127–147, 1998.

[ROY69] B. ROY. « Algèbre moderne et théorie des graphes », vol. 1, chapitre III, page 192. Henri Hierche, dunod édition, 1969.

[ROZENF88] A. ROZENFELD & A. SHER. « Detection and delineation of compact objects using intensity pyramids ». *Pattern Recognition*, vol. 21, pages 147–151, 1988.

[SHEN96] J. SHEN & D. SHEN. « Orthogonal Legendre Moments and Their Calculation ». Dans *ICPR'1996*, vol. 2, page 241, 1996.

[SMOLA03] A. J. SMOLA & R. KONDOR. « Kernels and Regularization on Graphs ». Dans M. WARMUTH & B. SCHÖKOPF, rédacteurs, *Proceedings of the Conference on Learning Theory and Kernels Workshop*, 2003.

[T.72A] W. T. & L. A.B. « Counting rooted maps by genus ». *Journal of Combinatorial Theory*, 1(13B), pages 192–218, 1972.

[T.72B] W. T. & L. A.B. « Counting rooted maps by genus ». *Journal of Combinatorial Theory*, 2(13B), pages 122–141, 1972.

[TANIMO75] S. L. TANIMOTO & T. PAVLIDIS. « A hierarchical data structure for picture processing ». *Computer Graphics and Image Processing*, vol. 4, pages 104–119, 1975.

[TARJAN75] R. E. TARJAN. « Efficiency of a good but non linear set union algorithm ». *J. Assoc. Comput. System Mach.*, vol. 22, pages 215–225, 1975.

[TUTTE68] W. T. TUTTE. « On the enumeration of planar maps ». *Bull. Amer. Math. Soc.*, vol. 74, pages 64–74, 1968.

[TUTTE73] W. T. TUTTE. « What is a map ? ». Dans *New directions in the theory of graphs (Proc. Third Ann Arbor Conf., Univ. Michigan, Ann Arbor, Mich., 1971)*, pages 309–325. Academic Press, New York, 1973.

[ULLMAN76] J. R. ULLMANN. « An Algorithm for Subgraph Isomorphism ». *J. ACM*, 23(1), pages 31–42, 1976. ISSN 0004-5411.

[VAUTRO96] P. VAUTROT, N. BONNET & M. HERBIN. « Comparative study of different spatial-frequency method (Gabor filters, wavelets, wavelet packets) for texture segmentation/classification ». Dans *IEEE International Conference Image Processing ICIP'96*, Lausanne, 1996.

[WALLAC99] C. WALLACE & D. DOWE. « Minimum message length and Kolmogorov complexity ». *Computer Journal (special issue on Komogorov complexity)*, 42(4), pages 270–283, 1999.

[WILF86] H. S. WILF. « Spectral bounds for the clique and independence numbers of graphs ». *J. Comb. Theory Ser. B*, 40(1), pages 113–117, 1986. ISSN 0095-8956.

[WILLER94] D. WILLERSINN & W. G. KROPATSCH. « Dual Graph Contraction for Irregular Pyramids ». Dans *International Conference on Pattern Recogntion D : Parallel Computing*, pages 251–256, International Association for Pattern Recognition, Jerusalem, Israel, 1994.

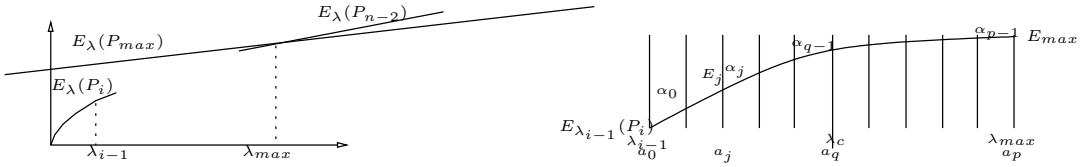
[WITKIN84] A. P. WITKIN. « Image Understanding », chapitre Scale Space Filtering : a new approach to multi-scale description, chapter 3, pages 79–95. Ablex, 1984.

[W.T.84] T. W.T. « Graph Theory », vol. 21. Encyclopedia of Mathematics, 1984.

ANNEXE

A.1 Interpolation de l'énergie

Dans cette section nous allons détailler les différentes étapes de calcul permettant d'obtenir une expression formelle de l'aire située sous la courbe $E_\lambda(C^*_\lambda(H))$, par la détermination de ϵ_1, ϵ_2 et de l'interpolation de $E_\lambda(C^*_\lambda(H))$ sur $[\lambda_{i-1}, \lambda_{max}]$, définie section 3.2.2.



(a) Construction de la hiérarchie (b) Interpolation

FIG. A.1 – Représentation des données connues lors de la construction de la hiérarchie (a) et illustrations des principales notations lors de l'interpolation(b).

Afin de simplifier les équations à venir nous oublions momentanément l'indice i et redéfinissons les notations illustrées sur la figures A.1 par les notations suivantes :

$$\begin{aligned} E_0 &= E_{\lambda_{i-1}}(P_i) \\ \lambda_0 &= \lambda_{i-1} \\ \alpha_0 &= C_{i-1} \\ \alpha_p &= C_{n-2} \\ E_{max} &= E_{\lambda_{max}}(P_{max}) \end{aligned}$$

où p désigne le nombre de régions de la partition courante.

Nous nous plaçons dans des conditions similaires à la section 3.2.2 en considérant l'intervalle $[\lambda_0, \lambda_{max}]$. Définissons les pentes à l'origine et à l'arrivée respectivement par α_0 (pente à l'origine) et α_p (pente à l'arrivée) et supposons qu'il nous reste exactement p fusions à effectuer avant d'obtenir la partition triviale ne comportant qu'une région.

Découpons l'intervalle $[\lambda_0, \lambda_{max}]$ en deux sous intervalles $[\lambda_0, \lambda_c] \cup [\lambda_c, \lambda_{max}]$. Où $\lambda_c \in [\lambda_0, \lambda_{max}]$ est une paramêtre de l'interpolation arbitrairement fixé à $\frac{\lambda_0+\lambda_{max}}{2}$. Chacun des deux sous intervalles $[\lambda_0, \lambda_c]$ et $[\lambda_c, \lambda_{max}]$ est découpé en $q = [\frac{p}{2}]$ sous intervalles de façon à obtenir p sous intervalles sur $[\lambda_0, \lambda_{max}]$. Supposons de plus que la largeur des q sous intervalles est fixe. On a donc :

$$\Delta_1 = \frac{\lambda_c - \lambda_0}{q} \text{ et } \Delta_2 = \frac{\lambda_m - \lambda_c}{q}$$

où Δ_1 et Δ_2 représentent respectivement la largeur des intervalles sur $[\lambda_0, \lambda_c]$ et $[\lambda_c, \lambda_{max}]$.

Chaque intervalle I_j, $j \in \{0, \ldots, p-1\}$ est affecté d'une pente α_j, α_0 et α_{p-1} étant fixés.

La courbe $E_\lambda(C^*_\lambda(H))$ étant concave et linéaire par morceaux, nous supposons de plus que les pentes α_j décroissent régulièrement de pas ϵ_1 et ϵ_2 sur chacun des intervalles $[\lambda_0, \lambda_c]$ et $[\lambda_c, \lambda_{max}]$. On a donc :

$$\begin{cases} \alpha_i &= \alpha_0 - i\epsilon_1, & \forall i \in \{0, \ldots, q-1\} \\ \alpha_i &= \alpha_{q-1} - (i-q+1)\epsilon_2, & \forall i \in \{q, \ldots, p-1\} \end{cases}$$

La première contrainte à imposer pour déterminer ϵ_1 et ϵ_2 et d'indiquer que la décroissace des pentes doit nous amener à α_{p-1} (connu). On a donc :

$$\alpha_{p-1} = \alpha_{q-1} - (p-q)\epsilon_2 = \alpha_0 - (q-1)\epsilon_1 - (p-q)\epsilon_2$$

Ce qui nous donne :

$$(q-1)\epsilon_1 + (p-q)\epsilon_2 = \alpha_0 - \alpha_{p-1} \tag{A.1}$$

La seconde contrainte et que partant de E_0 nous arrivions au bout des p intervalles à E_{max}. On a donc :

$$\begin{array}{lcl}
E_0 + \sum_{i=0}^{q-1} \Delta_1\alpha_i + \sum_{i=q}^{p-1} \Delta_2\alpha_i & = & E_{max} \\
\Delta_1 \sum_{i=0}^{q-1} \alpha_i + \Delta_2 \sum_{i=q}^{p-1} \alpha_i & = & E_{max} - E_0 \\
\Delta_1 \left(\sum_{i=0}^{q-1} \alpha_0 - i\epsilon_1\right) + \Delta_2 \left(\sum_{i=q}^{p-1} \alpha_{q-1} - (i-q+1)\epsilon_2\right) & = & E_{max} - E_0 \\
\Delta_1 q\alpha_0 - \Delta_1\epsilon_1 \sum_{i=0}^{q-1} i + \Delta_2 \left((p-q)\alpha_{q-1} - \epsilon_2 \sum_{i=q}^{p-1}(i-q+1)\right) & = & E_{max} - E_0 \\
\Delta_1 q\alpha_0 - \Delta_1\epsilon_1 \frac{q(q-1)}{2} + \Delta_2 \left((p-q)(\alpha_0 - (q-1)\epsilon_1) - \epsilon_2 \sum_{j=1}^{p-q} j\right) & = & E_{max} - E_0 \\
\Delta_1 q\alpha_0 + \Delta_2(p-q)\alpha_0 - \Delta_1\epsilon_1 \frac{q(q-1)}{2} - \Delta_2 \left((p-q)(q-1)\epsilon_1) + \epsilon_2 \frac{(p-q)(p-q+1)}{2}\right) & = & E_{max} - E_0
\end{array}$$

Ce qui nous donne :

$$\begin{array}{lcl}
\Delta_1\epsilon_1 \frac{q(q-1)}{2} + \Delta_2 \left((p-q)(q-1)\epsilon_1) + \epsilon_2 \frac{(p-q)(p-q+1)}{2}\right) & = & \Delta_1 q\alpha_0 + \Delta_2(p-q)\alpha_0 - (E_{max} - E_0) \\
\epsilon_1(q-1)\left(\Delta_1 \frac{q}{2} + \Delta_2(p-q)\right) + \epsilon_2\Delta_2 \frac{(p-q)(p-q+1)}{2} & = & \Delta_1 q\alpha_0 + \Delta_2(p-q)\alpha_0 - (E_{max} - E_0)
\end{array}$$

En utilisant notre première contrainte (équation A.1), il vient :

$$\epsilon_1(q-1)\left(\Delta_1\frac{q}{2} + \Delta_2(p-q)\right) + \Delta_2\frac{p-q+1}{2}(\alpha_0 - \alpha_{p-1} - (q-1)\epsilon_1) = \Delta_1 q\alpha_0 + \Delta_2(p-q)\alpha_0 - (E_{max} - E_0)$$

d'où :

$$\frac{1}{2}\epsilon_1(q-1)\left(\Delta_1 q + \Delta_2(p-q-1)\right) = \Delta_1 q\alpha_0 + \Delta_2(p-q)\alpha_0 - (E_{max} - E_0) - \Delta_2\frac{p-q+1}{2}(\alpha_0 - \alpha_{p-1})$$

Ce qui nous donne finalement :

$$\epsilon_1 = 2\frac{\Delta_1 q\alpha_0 + \Delta_2(p-q)\alpha_0 - (E_{max} - E_0) - \Delta_2\frac{p-q+1}{2}(\alpha_0 - \alpha_{p-1})}{(q-1)\left(\Delta_1 q + \Delta_2(p-q-1)\right)}$$

Le second epsilon (ϵ_2) se déduit facilement de ϵ_1 à partir de l'équation A.1. On a donc une courbe composée d'une série de segments de droites définis sur des intervalles (q

intervalles Δ_1, puis $p-q$ intervalles Δ_2). Chaque segment de droite sur un intervalle $I_i = [\lambda_i, \lambda_{i+1}]$, $i \in \{0, \ldots, p-1\}$ peut être défini par le segment de droite $E_i + \alpha_i(\lambda - \lambda_i)$ où E_i est la valeur de la courbe (et donc de la droite) en λ_i.

Définissons les différents E_i :

– Pour $i \in \{0, \ldots, q\}$, on a $D_0 = E_0$ et pour tout i dans $\{1, \ldots, q\}$

$$\begin{aligned} E_i &= E_0 + \textstyle\sum_{j=0}^{i-1} \alpha_j \Delta_1 \\ &= E_0 + \Delta_1 \textstyle\sum_{j=0}^{i-1} \alpha_j \\ &= E_0 + \Delta_1 \textstyle\sum_{j=0}^{i-1} \alpha_0 - j\epsilon_1 \\ &= E_0 + i\Delta_1\alpha_0 - \Delta_1\epsilon_1 \frac{i(i-1)}{2} \\ &= E_0 + i\Delta_1 \left(\alpha_0 + \tfrac{1}{2}\epsilon_1\right) - \tfrac{i^2}{2}\Delta_1\epsilon_1 \end{aligned}$$

Notons qu'en utilisant la formule précédente on obtient bien $D_0 = E_0$. Celle-ci est donc valable pour tout $i \in \{0, \ldots, q\}$.

– Pour $i \in \{q, \ldots, p-1\}$, on a $E_q = E_0 + q\Delta_1 \left(\alpha_0 + \frac{1}{2}\epsilon_1\right) - \frac{q^2}{2}\Delta_1\epsilon_1$ et pour tout i dans $\{q+1, \ldots, p-1\}$:

$$\begin{aligned} E_i &= E_q + \textstyle\sum_{j=q}^{i-1} \alpha_j \Delta_2 \\ &= E_q + \Delta_2 \textstyle\sum_{j=q}^{i-1} \alpha_{q-1} - (j-q+1)\epsilon_2 \\ &= E_q + (i-q)\Delta_2\alpha_{q-1} - \epsilon_2\Delta_2 \textstyle\sum_{j=q}^{i-1} j - q + 1 \\ &= E_q + (i-q)\Delta_2\alpha_{q-1} - \epsilon_2\Delta_2 \textstyle\sum_{k=1}^{i-q} k \\ &= E_q + (i-q)\Delta_2\alpha_{q-1} - \epsilon_2\Delta_2 \frac{(i-q)(i-q+1)}{2} \\ &= E_q - q\Delta_2\alpha_{q-1} - \tfrac{1}{2}\epsilon_2\Delta_2 q(q-1) + i\Delta_2 \left(\alpha_{q-1} + \tfrac{1}{2}\epsilon_2(2q-1)\right) - \tfrac{i^2}{2}\epsilon_2\Delta_2 \end{aligned}$$

Notons qu'en utilisant la formule précédente nous avons bien $E_i = E_q$ pour $i = q$. La formule est donc valable pour tout $i \in \{q, \ldots, p-1\}$.

Les λ_i se calculent aisément à l'aide des formules suivantes :

$$\begin{cases} \lambda_i &= \lambda_0 + i\Delta_1, & \forall i \in \{0, \ldots, q\} \\ \lambda_i &= \lambda_q + (i-q)\Delta_2, & \forall i \in \{q, \ldots, p-1\} \end{cases}$$

En résumé nous avons donc :

$$\begin{aligned} &\forall i \in \{0, \ldots, q-1\} \\ &\qquad \begin{cases} \alpha_i &= \alpha_0 - i\epsilon_1 \\ \lambda_i &= \lambda_0 + i\Delta_1 \\ E_i &= E_0 + i\Delta_1 \left(\alpha_0 + \frac{1}{2}\epsilon_1\right) - \frac{i^2}{2}\Delta_1\epsilon_1 \end{cases} \\ &\forall i \in \{q, \ldots, p-1\} \\ &\qquad \begin{cases} \alpha_i &= \alpha_{q-1} - (i-q+1)\epsilon_2 \\ \lambda_i &= \lambda_q + (i-q)\Delta_2 \\ E_i &= E_q - q\Delta_2\alpha_{q-1} - \frac{1}{2}\epsilon_2\Delta_2 q(q-1) \\ &\quad\ + i\Delta_2 \left(\alpha_{q-1} + \frac{1}{2}\epsilon_2(2q-1)\right) - \frac{i^2}{2}\epsilon_2\Delta_2 \end{cases} \end{aligned} \tag{A.2}$$

Nous avons donc tous les paramêtres de la courbe interpolant $E_\lambda(C^*_\lambda(H))$ entre λ_0 et λ_{max}. On peut donc à présent calculer l'aire sous la courbe interpolée.

A.1.1 Calcul de l'intégrale

L'intégrale représentant l'aire sous la courbe (voir Fig 3.4) peut être définie à partir de l'équation suivante :

$$\sum_{i=0}^{p-1}\int_{\lambda_i}^{\lambda_{i+1}} E_i+\alpha_i(\lambda-\lambda_i)d\lambda = \sum_{i=0}^{q-1}\int_{\lambda_i}^{\lambda_{i+1}} E_i+\alpha_i(\lambda-\lambda_i)d\lambda+\sum_{i=q}^{p-1}\int_{\lambda_i}^{\lambda_{i+1}} E_i+\alpha_i(\lambda-\lambda_i)d\lambda \tag{A.3}$$

Nous allons dans un premier temps étudier le premier terme de l'équation A.3, représentant la somme de 0 à $q-1$:

$$\begin{aligned}
\sum_{i=0}^{q-1}\int_{\lambda_i}^{\lambda_{i+1}} E_i+\alpha_i(\lambda-\lambda_i)d\lambda &= \sum_{i=0}^{q-1}(E_i-\alpha_i\lambda_i)\Delta_1+\frac{1}{2}\alpha_i\Delta_1(\lambda_{i+1}+\lambda_i)\\
&= \Delta_1\sum_{i=0}^{q-1}E_i-\alpha_i\lambda_i+\frac{1}{2}\alpha_i\lambda_{i+1}+\frac{1}{2}\alpha_i\lambda_i\\
&= \Delta_1\sum_{i=0}^{q-1}E_i+\frac{1}{2}\alpha_i\Delta_1\\
&= \Delta_1\sum_{i=0}^{q-1}E_i+\frac{1}{2}\Delta_1^2\sum_{i=0}^{q-1}\alpha_i
\end{aligned}$$

Nous obtenons alors une sommet de deux termes. Calculons tout d'abord son premier terme, à savoir la somme des E_i (équation A.2) :

$$\begin{aligned}
\sum_{i=0}^{q-1}E_i &= \sum_{i=0}^{q-1}E_0+i\Delta_1\left(\alpha_0+\frac{1}{2}\epsilon_1\right)-\frac{i^2}{2}\Delta_1\epsilon_1\\
&= qE_0+\Delta_1\left(\alpha_0+\tfrac{1}{2}\epsilon_1\right)\sum_{i=0}^{q-1}i-\frac{1}{2}\Delta_1\epsilon_1\sum_{i=0}^{q-1}i^2\\
&= qE_0+\Delta_1\left(\alpha_0+\tfrac{1}{2}\epsilon_1\right)\tfrac{q(q-1)}{2}-\Delta_1\epsilon_1\tfrac{q(q-1)(2q-1)}{12}
\end{aligned}$$

Calculons ensuite son second terme, représentant la somme des α_i :

$$\begin{aligned}
\sum_{i=0}^{q-1}\alpha_i &= \sum_{i=0}^{q-1}\alpha_0-i\epsilon_1\\
&= \alpha_0 q-\epsilon_1\sum_{i=0}^{q-1}i\\
&= \alpha_0 q-\epsilon_1\tfrac{q(q-1)}{2}
\end{aligned}$$

Nous obtenons ainsi pour la somme allant de 0 à $q-1$:

$$\sum_{i=0}^{q-1}\int_{\lambda_i}^{\lambda_{i+1}} E_i+\alpha_i(\lambda-\lambda_i)d\lambda = q\Delta_1E_0+\frac{1}{2}\alpha_0\Delta_1^2q^2-\Delta_1^2\epsilon_1\frac{q(q-1)(2q-1)}{12} \tag{A.4}$$

Calculons à présent le second terme de l'équation A.3, représentant la somme allant de q à $p-1$.

En s'inspirant de la démonstration précédente nous trouvons facilement :

$$\sum_{i=q}^{p-1}\int_{\lambda_i}^{\lambda_{i+1}} E_i+\alpha_i(\lambda-\lambda_i)d\lambda = \Delta_2\sum_{i=q}^{p-1}E_i+\frac{1}{2}\Delta_2^2\sum_{i=q}^{p-1}\alpha_i$$

Calculons dans un premier temps la somme des E_i (équation A.2) :

$$\begin{aligned}\sum_{i=q}^{p-1}E_i &= \textstyle\sum_{i=q}^{p-1}E_q - q\Delta_2\alpha_{q-1} - \frac{1}{2}\epsilon_2\Delta_2 q(q-1) + i\Delta_2\left(\alpha_{q-1}+\frac{1}{2}\epsilon_2(2q-1)\right) - \frac{i^2}{2}\epsilon_2\Delta_2\\ &= (p-q)\left[E_q - q\Delta_2\alpha_{q-1} - \tfrac{1}{2}\epsilon_2\Delta_2 q(q-1)\right]\\ &\quad +\Delta_2\left(\alpha_{q-1}+\tfrac{1}{2}\epsilon_2(2q-1)\right)\textstyle\sum_{i=q}^{p-1} i\\ &\quad -\tfrac{1}{2}\epsilon_2\Delta_2\textstyle\sum_{i=q}^{p-1} i^2\\ &= (p-q)\left[E_q - q\Delta_2\alpha_{q-1} - \tfrac{1}{2}\epsilon_2\Delta_2 q(q-1)\right]\\ &\quad +\tfrac{1}{2}\Delta_2\left[\alpha_{q-1}+\tfrac{1}{2}\epsilon_2(2q-1)\right][n(p-1)-q(q-1)]\\ &\quad -\tfrac{1}{12}\epsilon_2\Delta_2[n(p-1)(2p-1)-q(q-1)(2q-1)]\end{aligned}$$

Calculons ensuite la somme des α_i :

$$\begin{aligned}\sum_{i=q}^{p-1}\alpha_i &= \sum_{i=q}^{p-1}\alpha_{q-1}-(i-q+1)\epsilon_2\\ &= (p-q)\alpha_{q-1}-\epsilon_2\sum_{i=q}^{p-1} i-q+1\\ &= (p-q)\alpha_{q-1}-\epsilon_2\sum_{j=1}^{p-q} j\\ &= (p-q)\alpha_{q-1}-\epsilon_2\tfrac{(p-q)(p-q+1)}{2}\end{aligned}$$

Nous obtenons ainsi pour la somme allant de q à $p-1$ l'expression suivante :

$$\sum_{i=q}^{p-1}\int_{\lambda_i}^{\lambda_{i+1}} E_i+\alpha_i(\lambda-\lambda_i)d\lambda = (p-q)\Delta_2 E_q+\frac{1}{2}\Delta_2^2(p-q)^2\alpha_{q-1}-\frac{1}{12}(p-q)\epsilon_2\Delta_2^2\left[2(p-q)^2+3(p-q)+1\right] \tag{A.5}$$

En sommant les équations A.4 et A.5 nous obtenons finalement l'expression de l'intégrale que nous cherchons :

$$\begin{aligned}\mathcal{A}=\sum_{i=0}^{p-1}\int_{\lambda_i}^{\lambda_{i+1}} E_i+\alpha_i(\lambda-\lambda_i)d\lambda &= q\Delta_1 E_0+\tfrac{1}{2}\alpha_0\Delta_1^2q^2-\Delta_1^2\epsilon_1\tfrac{q(q-1)(2q-1)}{12}\\ &\quad +(p-q)\Delta_2E_q+\tfrac{1}{2}\Delta_2^2(p-q)^2\alpha_{q-1}\\ &\quad -\tfrac{1}{12}(p-q)\epsilon_2\Delta_2^2[2(p-q)^2+3(p-q)+1]\end{aligned}$$

$$avec \begin{cases} E_q = E_0 + q\Delta_1 \left(\alpha_0 + \frac{1}{2}\epsilon_1\right) - \frac{q^2}{2}\Delta_1\epsilon_1 & \text{(valeur de la courbe interpolée en } \lambda_c) \\ \Delta_1 = \frac{\lambda_c - \lambda_{i-1}}{q} & \text{(largeur des intervalles } I_j \text{ sur } [\lambda_{i-1}, \lambda_c]) \\ \Delta_2 = \frac{\lambda_{max} - \lambda_c}{q} & \text{(largeur des intervalles } I_j \text{ sur } [\lambda_c, \lambda_{max}]) \end{cases}$$

Nous obtenons ainsi une expression formelle de l'intégrale en fonction,de $n, i, \lambda_0, \lambda_{max}$, E_0, E_{max} qui sont supposés être des données connues du problème.

A.2 Démonstration de la proposition 3

Comme nous l'avons vu dans la section 5.1 l'étape qui consiste à réduire le nombre de couples de sommets candidats à l'appariement, en sélectionnant l'ensemble des couples $(v, v') \in \mathcal{V}_P \times \mathcal{V}_{P'}$ tels que

$$\Delta_{F_1}(v, v') = |1 - \frac{Feat_1(v')}{Feat_1(v)}| \leq \epsilon_1, \tag{A.6}$$

revient à sélectionner tous les sommets v' de $\mathcal{V}_{P'}$ dont la première composante du vecteur de caractéristiques est comprise dans l'intervalle suivant :

$$\begin{array}{l} Feat_1(v') \in [(1-\epsilon_1)Feat_1(v), (1+\epsilon_1)Feat_1(v)] \text{ si } Feat_1(v) > 0 \\ Feat_1(v') \in [(1+\epsilon_1)Feat_1(v), (1-\epsilon_1)Feat_1(v)] \text{ si } Feat_1(v) < 0 \end{array} \tag{A.7}$$

Supposons que la caractéristique $Feat_1(v)$ soit strictement positive ($\forall v \in \mathcal{V}_P \quad Feat_1(v) > 0$). Nous avons donc dans le cas où la première caractéristique du sommet v' et légerement inférieure à celle du sommet v :

$$\begin{array}{l} \text{si } (1-\epsilon_1)Feat_1(v) \leq Feat_1(v') \leq Feat_1(v) \\ \quad \Leftrightarrow (1-\epsilon_1) \leq \frac{Feat_1(v')}{Feat_1(v)} \leq 1 \\ \quad \Leftrightarrow 0 \leq 1 - \frac{Feat_1(v')}{Feat_1(v)} \leq \epsilon_1 \end{array} \tag{A.8}$$

et de la même manière, dans le cas ou la première caractéristique du sommet v' et légerement supérieure à celle du sommet v :

$$\begin{array}{l} \text{si } Feat_1(v) \leq Feat_1(v') \leq (1+\epsilon_1)Feat_1(v) \\ \quad \Leftrightarrow \frac{1}{1+\epsilon_1} \leq \frac{Feat_1(v)}{Feat_1(v')} \leq 1 \\ \quad \Leftrightarrow 0 \leq 1 - \frac{Feat_1(v)}{Feat_1(v')} \leq 1 - \frac{1}{1+\epsilon_1} \end{array} \tag{A.9}$$

or $1 - \frac{1}{1+\epsilon_1} = \frac{\epsilon_1}{1+\epsilon_1}$ et $\frac{\epsilon_1}{1+\epsilon_1} \leq \epsilon_1$

Nous obtenons finalement :

$$0 \leq 1 - \frac{Feat_1(v)}{Feat_1(v')} \leq \epsilon_1 \tag{A.10}$$

Ce qui nous donne immédiatement l'implication suivante :

$$\Delta_{F_1}(v, v') \leq \epsilon_1 : f_1(v, v') \leq \epsilon_1 \tag{A.11}$$

La validité de la condition 5.7 ($\Delta_{F_1}(v, v') \leq \epsilon_1$) implique donc forcement que la condition 5.8 ($f_1(v, v') \leq \epsilon_1$) est également vérifiée.

A.3 Calcul des moments de Legendre (jusqu'à l'ordre 8)

Nous alons détailler dans cette section, les différents calculent nécessaire à l'obtention des moments de Legendre jusqu'à l'ordre 8.

– Il faut tout d'abord commencer par calculer les moments cumulés définis par :

$$S_i(0) = S_{i-1}(0)$$

$$S_i(j) = S_i(j-1) + S_{i-1}(j)$$

– calculer ensuite $< L_0, S_i >= S_{i+1}(x+k) - S_{i+1}(x-k)$
– puis calculer

$$< L_1, S_i >= S_{i+1}(x+k) + S_{i+1}(x-k) - \frac{1}{k} < L_0, S_{i+1} >$$

– finalement calculer les différents moments :

1. $M_0 =< L_0, S >$
2. $M_1 =< L_1, S >$
3. M_2

$$\begin{aligned} M_2 &= < L_2, S > \\ &= < L_0, S > - \tfrac{3}{k} < L_1, S_1 > \end{aligned}$$

4. M_3

$$\begin{aligned} M_3 &= < L_3, S > \\ &= < L_1, S > - \tfrac{5}{k} < L_2, S_1 > \\ &= < L_1, S > - \tfrac{5}{k} \left(< L_0, S_1 > - \tfrac{3}{k} < L_1, S_2 > \right) \\ &= < L_1, S > - \tfrac{5}{k} < L_0, S_1 > + \tfrac{15}{k^2} < L_1, S_2 > \end{aligned}$$

5. M_4

$$\begin{aligned} M_4 &= < L_4, S > \\ &= < L_2, S > -\tfrac{7}{k} < L_3, S_1 > \\ &= < L_0, S > -\tfrac{3}{k} < L_1, S_1 > -\tfrac{7}{k}\left(< L_1, S_1 > -\tfrac{5}{k} < L_2, S_2 >\right) \\ &= < L_0, S > -\tfrac{10}{k} < L_1, S_1 > +\tfrac{35}{k^2}\left(< L_0, S_2 > -\tfrac{3}{k} < L_1, S_3 >\right) \\ &= < L_0, S > -\tfrac{10}{k} < L_1, S_1 > +\tfrac{35}{k^2} < L_0, S_2 > -\tfrac{105}{k^3} < L_1, S_3 > \end{aligned}$$

6. M_5

$$\begin{aligned} M_5 &= < L_5, S > \\ &= < L_3, S > -\tfrac{9}{k} < L_4, S_1 > \\ &= < L_1, S > -\tfrac{5}{k} < L_2, S_1 > -\tfrac{9}{k}\left(< L_2, S_1 > -\tfrac{7}{k} < L_3, S_2 >\right) \\ &= < L_1, S > -\tfrac{14}{k} < L_2, S_1 > +\tfrac{63}{k^2} < L_3, S_2 > \\ &= < L_1, S > -\tfrac{14}{k}\left(< L_0, S_1 > -\tfrac{3}{k} < L_1, S_2 >\right) + \tfrac{63}{k^2}\left(< L_1, S_2 > -\tfrac{5}{k} < L_2, S_3 >\right) \\ &= < L_1, S > -\tfrac{14}{k} < L_0, S_1 > +\tfrac{42}{k^2} < L_1, S_2 > +\tfrac{63}{k^2} < L_1, S_2 > -\tfrac{315}{k^3} < L_2, S_3 > \\ &= < L_1, S > -\tfrac{14}{k} < L_0, S_1 > +\tfrac{105}{k^2} < L_1, S_2 > -\tfrac{315}{k^3}\left(< L_0, S_3 > -\tfrac{3}{k} < L_1, S_4 >\right) \\ &= < L_1, S > -\tfrac{14}{k} < L_0, S_1 > +\tfrac{105}{k^2} < L_1, S_2 > -\tfrac{315}{k^3} < L_0, S_3 > +\tfrac{945}{k^4} < L_1, S_4 > \end{aligned}$$

7. M_6

$$\begin{aligned} M_6 &= < L_6, S > \\ &= < L_4, S > -\tfrac{11}{k} < L_5, S_1 > \end{aligned}$$

Calcul de $< L_4, S >$

$$\begin{aligned} < L_4, S > &= < L_2, S > -\tfrac{7}{k} < L_3, S_1 > \\ &= < L_0, S > -\tfrac{3}{k} < L_1, S_1 > -\tfrac{7}{k}\left(< L_1, S_1 > -\tfrac{5}{k} < L_2, S_2 >\right) \\ &= < L_0, S > -\tfrac{10}{k} < L_1, S_1 > +\tfrac{35}{k^2} < L_2, S_2 > \\ &= < L_0, S > -\tfrac{10}{k} < L_1, S_1 > +\tfrac{35}{k^2}\left(< L_0, S_2 > -\tfrac{3}{k} < L_1, S_3 >\right) \\ &= < L_0, S > -\tfrac{10}{k} < L_1, S_1 > +\tfrac{35}{k^2} < L_0, S_2 > -\tfrac{105}{k^3} < L_1, S_3 > \end{aligned}$$

Calcul de $< L_5, S_1 >$:

$$\begin{aligned}
< L_5, S_1 > &= < L_3, S_1 > -\tfrac{9}{k} < L_4, S_2 > \\
&= < L_1, S_1 > -\tfrac{5}{k} < L_2, S_2 > -\tfrac{9}{k}\left(< L_2, S_2 > -\tfrac{7}{k} < L_3, S_3 >\right) \\
&= < L_1, S_1 > -\tfrac{14}{k} < L_2, S_2 > +\tfrac{63}{k^2} < L_3, S_3 > \\
&= < L_1, S_1 > -\tfrac{14}{k}\left(< L_0, S_2 > -\tfrac{3}{k} < L_1, S_3 >\right) + \tfrac{63}{k^2}\left(< L_1, S_3 > -\tfrac{5}{k} < L_2, S_4 >\right) \\
&= < L_1, S_1 > -\tfrac{14}{k} < L_0, S_2 > +\tfrac{105}{k^2} < L_1, S_3 > -\tfrac{315}{k^3} < L_2, S_4 > \\
&= < L_1, S_1 > -\tfrac{14}{k} < L_0, S_2 > +\tfrac{105}{k^2} < L_1, S_3 > -\tfrac{315}{k^3}\left(< L_0, S_4 > -\tfrac{3}{k} < L_1, S_5 >\right) \\
&= < L_1, S_1 > -\tfrac{14}{k} < L_0, S_2 > +\tfrac{105}{k^2} < L_1, S_3 > -\tfrac{315}{k^3} < L_0, S_4 > +\tfrac{945}{k^4} < L_1, S_5 >
\end{aligned}$$

Nous obtenons ainsi :

$$\begin{aligned}
M_6 = \; & < L_0, S > -\tfrac{21}{k} < L_1, S_1 > +\tfrac{189}{k^2} < L_0, S_2 > -\tfrac{1260}{k^3} < L_1, S_3 > \\
& +\tfrac{3465}{k^4} < L_0, S_4 > +\tfrac{10395}{k^5} < L_1, S_5 >
\end{aligned}$$

8. M_7 nous obtenons :

$$\begin{aligned}
M_7 = \; & < L_1, S > +\tfrac{1}{k} < L_0, S_1 > +\tfrac{168}{k^2} < L_1, S_2 > +\tfrac{2142}{k^3} < L_0, S_3 > \\
& -\tfrac{15435}{k^4} < L_1, S_4 > +\tfrac{45045}{k^5} < L_0, S_5 > -\tfrac{135135}{k^6} < L_1, S_6 >
\end{aligned}$$

9. M_8 nous obtenons :

$$\begin{aligned}
M_8 = \; & < L_0, S > +\tfrac{6}{k} < L_1, S_1 > -\tfrac{216}{k^2} < L_0, S_2 > -\tfrac{4410}{k^3} < L_1, S_3 > \\
& -\tfrac{38115}{k^4} < L_0, S_4 > -\tfrac{249480}{k^5} < L_1, S_5 > -\tfrac{675675}{k^6} < L_0, S_6 > \\
& -\tfrac{2027025}{k^7} < L_1, S_7 >
\end{aligned}$$

A.3.1 Calcul des polynomes de Legendre (jusqu'à l'ordre 6)

Pour calculer les moments de Legendre jusqu'à l'ordre 8, il est nécessaire de calculer au préalable les polynômes de Legendre jusqu'à l'ordre 6. La formule générale est donnée par l'expression suivante :

$$L_n(x) = \frac{d^n}{dx^n}\frac{(x^2-k^2)^n}{(2k)^n.n!}$$

Les polynômes de Legendre de l'ordre 0 à 6 sont alors :

$$L_0(x) = 1$$

$$L_1(x) = \frac{d}{dx}\frac{(x^2-k^2)}{(2k)} = \frac{x}{k}$$

$$L_2(x) = \frac{d^2}{dx^2}\frac{(x^2-k^2)^2}{(2k)^2.2!} = \frac{3x^2-k^2}{2k^2}$$

$$L_3(x) = \frac{d^3}{dx^3}\frac{(x^2-k^2)^3}{(2k)^3.3!} = \frac{5x^3-3k^2x}{3k^3}$$

$$L_4(x) = \frac{d^4}{dx^4}\frac{(x^2-k^2)^4}{(2k)^4.4!} = \frac{35x^4-30k^2x^2+3k^4}{8k^4}$$

$$L_5(x) = \frac{d^5}{dx^5}\frac{(x^2-k^2)^5}{(2k)^5.5!} = \frac{63x^5-70k^2x^3+15k^4x}{8k^5}$$

$$L_6(x) = \frac{d^6}{dx^6}\frac{(x^2-k^2)^6}{(2k)^6.6!} = \frac{231x^6-315k^2x^4+105k^4x^2-5k^6}{16k^6}$$

A.4 Algorithme de Dijkstra

L'algorithme de Dijkstra présenté sur l'algorithme 6 permet de calculer le plus court chemin menant d'un sommet de départ s à tous les autres sommets du graphe en faisant grossir un ensemble de sommet S à partir duquel il peut calculer le plus court chemin. A chaque étape de l'algorithme le sommet a ajouter à l'ensemble S est déterminé à l'aide d'une file de priorité Q qui contient l'ensemble des sommet de $V \setminus S$ classés selon une distance correspondant à la longueur du plus court chemin les séparant du sommet de départ. Le sommet u extrait de la pile est ajouté à l'ensemble S et toutes les arêtes qui en sortent sont pris en compte : si la distance de s à u ajoutée au poids de l'arête sortant (u, v) est inférieure à celle déja enregistrée pour le sommet v, alors la distance au sommet v est mise a jour. L'algorithme itère ce processus jusqu'à ce que la file Q soit vide.

Les sommets peuvent être étiquetés par trois états différents : (i) visité, (ii) non visté ou (iii) traité.Les sommets *non visités* correspondent aux sommets qui n'ont pas encore été parcourus, les *visités* sont ceux appartenant à l'ensemble $V \setminus S$ et les sommets *traités* sont ceux qui ont été mis dans S.

Algorithme 7 Algorithme de Dijkstra permettant de calculer le plus court chemin entre un sommets d'un graphe et les autres.

entrée : un graphe $G = (V, E)$**, un sommet de départ** s **et une fonction de point sur les arêtes** w
$Q = \emptyset, S = \emptyset$
for tout sommet u dans V **do**
  $dist[u] = \infty$
  $prec[u] = u$
  $etat[u] = \neg$visité
end for
$etat[s] =$ visité
$dist[s] = 0$
inserer s dans Q
while Q n'est pas vide **do**
  u = ExtraireMin(Q)
  $S = S \cup \{u\}$
  for tout sommet v voisin de u **do**
    if $w(u, v) + dist[u] < d[v]$ **then**
      $dist[v] := w(u, v) + dist[u]$
      $prec[v] := u$
      if $etat[v] = \neg$visité **then**
        $etat[v] =$ visité
        insérer v dans Q
      else
        if $etat[v] =$ visité **then**
          mettre à jour la valeur du sommet v dans Q
        end if
      end if
    end if
    $dist[u] = \infty$
    $prec[u] = u$
    $etat[u] = \neg$visité
  end for

end while
renvoyer $(dist, prec)$

RÉSUMÉ. Cette thèse utlise les pyramides combinatoires pour construire des hiérarchies de partitions et apparier des objets communs à celles-ci. Il se décompose donc en deux parties :

La première partie concerne la construction et la représentation de hiérarchies de partitions d'images couleur. Dans ce contexte, nous présentons un cadre formel basé sur un modèle énergétique. Ce cadre permet la construction de hiérarchies stables et robustes en réalisant des coupes optimales dans des hiérarchies indicées. Nous proposons et évaluons différentes heuristiques de construction permettant d'obtenir ces hiérarchies.

La seconde partie traite d'appariement structurel hiérarchique. Nous partons du principe que les images présentent, en général, des structures cohérentes à différentes échelles. Une structure commune à deux scènes pourra voir sa représentation différer suivant l'échelle d'observation. Ainsi, nous proposons une méthode d'appariement hiérarchique utilisant plusieurs échelles d'observation. Cette méthode est basée sur une distance d'édition originale permettant de retrouver des structures communes à deux hiérarchies de partitions.

TITLE. Segmentation and hierarchical matching based on combinatorial pyramids framework

ABSTRACT. This thesis deals with the problems of segmentation and hierarchical matching based on combinatorial pyramids framework. It is organized into two parts :

The first part deals with the construction and the encoding of hierarchies of color images partitions. In this context, we propose a formal framework based on an energetic model. This framework builts stable and robust hierarchies by doing optimal cuts within indexed hierarchies. We propose and evaluate different construction schemes to obtain such hierarchies.

In a second part, we study the matching of different objects between two hierarchies of partitions. We start from the assumption that images usually present coherent structures at different scales. A common structure from two different scenes can have different representations depending from the scale of observation. We thus propose a hierarchical matching method using several scales of observation. This method is based on an original edit distance which allows to determine common structures within two hierarchies of partitions.

MOTS-CLÉS. Traitement d'images ; Graphes, Théorie des ; Hiérarchies ; Algorithmes ; Traitement d'images – Techniques numériques ; Segmentation ; Appariement ; Pyramide combinatoire.

KEYWORDS. Image processing ; Graph theory ; Hierarchies ; Algorithms ; Image processing – Digital techniques ; Segmentation ; Matching ;Combinatorial pyramid.

DISCIPLINE. Informatique – Traitement d'images.

GREYC–CNRS UMR 6072, Équipe Image
ENSICAEN, 6 Bd du Maréchal Juin, 14050 Caen Cedex France.