République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Dr Yahia Farès de Médéa
Faculté des Sciences et de la Technologie
Département du Génie Électrique et de l'Informatique



Mémoire

présenté en vue d'obtention du diplôme de Magister

Spécialité : Électronique

Option : Instrumentation et Microélectronique

Présenté par : BENIMAM Meriem

Thème

Segmentation d'images médicales par coupure de graphe

Soutenu publiquement le 04/09/2013 devant le jury :

Pr. Salem ABUDURA	Professeur, U. Médéa	Président
Dr. Noureddine BATEL	MCA, U. Médéa	Examinateur
Dr. Nadjia BENBLIDIA	MCA, U. Blida	Examinatrice
Pr. Abderrezak GUESSOUM	Professeur, U. Blida	Encadreur

REMERCIEMENTS

Je tiens tout d'abord à remercier mes profondes gratitudes à DIEU le tout puissant.

Je remercie Monsieur Abderrezak Guessoum, Professeur à l'université de Blida, pour la confiance qu'il m'a accordée en ce sujet. Qu'il soit remercie pour avoir dirigé mes travaux et pour en être l'encadreur de ce mémoire. Durant ces années de formation de post-graduation, j'ai surtout apprécié sa grande disponibilité, sa compétence scientifique, ses conseils et suggestions qui ont contribué à l'avancement de mon travail.

Mes remerciements vont également à Monsieur Salem ABUDURA, Professeur à l'Université de Médéa pour m'avoir fait l'honneur de présider le jury de ce mémoire.

Mes remerciements vont pareillement à Monsieur Noureddine BATEL, Maître de conférences à l'Université de Médéa et Madame Nadjia BENBLIDIA, Maître de conférences à l'Université de Blida pour m'avoir fait l'honneur de juger ce mémoire.

Mes remerciements s'adressent également à tous les enseignants qui ont contribué à ma formation de Magister.

Résumé: Dans ce mémoire, nous nous intéressons à la segmentation d'objets d'intérêt contenus dans les images médicales en utilisant la méthode de coupe de graphes. Afin de simplifier la problématique d'étiquetage de l'objet à segmenter et obtenir rapidement le résultat de la segmentation, nous nous sommes appuyés sur le graphe coupé de Boykov-Jolly qui, à travers leur modèle, permet de résoudre pour beaucoup d'utilisateurs la problématique de découpe d'images à des fins personnelles ou professionnelles. La procédure de découpage de graphes est organisée autour de la minimisation de la fonction d'énergie par l'intermédiaire de l'algorithme Flot Max/Coupe Min.

Mots clés: champ aléatoire de Markov, coupe minimale, étiquetage, flux maximal, graphe, segmentation d'images.

Abstract: In this work, we are interesting on interest objects segmentation contained in the medical images using the graph cuts method. In order to simplify the object labelling problem and to obtain rapidly the segmentation, we have recourse to the Boykow-Jolly graph cut model which can solve the cutting image problem aiming personal or professional activity. The graph cutting process is achieved using energy function minimization by Max Flow/Min Cut algorithm.

Key words: graph, image segmentation, labeling, Markov random field, max flow, min cut.

ملخص : يهدف هذا المشروع إلى تقسيم الأجزاء ذات الاهتمام الواردة في الصور الطبية باستخدام طريقة قطع المنحنيات. و لتبسيط مشكلة وصف الجزء الذي يهم والحصول على التقسيم السريع للتجزئة، اعتمدنا في هذا العمل على طريقة قطع المنحنيات المقترحة من طرف Boykov وJolly . والذي من خلال نموذجهما، نستطيع حل العديد من مشاكل قطع الصور لأغراض شخصية أو مهنية. ويتم تنفيذ قطع المنحنيات عن طريق خفض دالة الطاقة من خلال استعمال خوارزمية Max Flow/Min Cut .

كلمات البحث: المنحنيات، القطع الأدنى ، التدفق الأقصى، تجزئة صورة، وضع العلامات ، حقل ماركوف العشوائي.

Table des matières

Introduction générale……..…….. 1

Chapitre 1: Champ de Markov en traitement d'images

1.1 Introduction……. 4

1.2 Description et modélisation probabiliste de l'image…… 4

1.2.1 Description de l'image……... 4

1.2.2 Modélisation probabiliste de l'image……. 6

1.3 Champs de Markov - Champs de Gibbs…….. 6

1.3.1 Définition d'un champ de Markov……..… 6

1.3.2 Equivalence champs de Markov - champs de Gibbs……. 7

1.4 Quelques MRF fondamentaux…….…… 10

1.5 Exemple d'application : la segmentation……..… 12

1.6 Estimateur dans un cadre markovien…….. 14

1.6.1 Introduction……... 14

1.6.2 Modélisation bayésienne et fonction de coût……. 14

a) L'estimateur MAP……..…….…… 15

b) Solution algorithmique de l'estimateur MAP dans le cas markovien……. 16

1.7 Conclusion……... 17

Chapitre 2 : Généralités sur les graphes

2.1 Introduction……. 18

2.2 Les ensembles……..... 18

2.3 Graphes orientés et non-orientés…… 19

2.3.1 Graphes orientés…… 19

2.3.1.1 Graphe réflexif…… 21

2.3.1.2 Graphe symétrique et fermeture symétrique…….. 21

2.3.2 Graphes non-orientés……. 23

2.3.3 Association entre graphes orientés et graphes non-orientés…….. 24

2.4 Notions communes et relations entre graphes orientés et non-orientés…….. 24

2.4.1 Sous-graphe……... 25

2.4.2 Notations ensemblistes appliquées aux graphes…… 25

2.4.3 Connexité et composantes connexes…… 25
2.4.4 Complémentaire dans un graphe……... 27
2.4.5 Adjacence et fonction successeur…….. 27
2.4.6 Clique et graphe complet…….. 28
2.4.7 Graphes isomorphes…….. 29
2.4.8 Graphe dérivé…… 29
2.4.8.1 Graphe dérivé non-orienté…….. 29
2.4.8.2 Graphe dérivé orienté…… 32
2.4.8.3 Notations et remarques sur les graphes dérivés orientés ou non…….. 34
2.4.9 Poids sur un graphe…… 35
2.5 Graphes usuels en traitement d'images……... 38
2.5.1 Qu'est-ce qu'une image numérique ?..... 38
2.5.2 Voisinage dans une image……. 38
2.5.3 Image, voisinage et graphe……... 40
2.6 Conclusion……... 41

Chapitre 3 : Coupe minimale et flot maximal
3.1 Introduction……. 42
3.2 Réseau de transport et flot……... 42
3.2.1 Réseau de transport…… 42
3.2.2 Flot…… 43
3.3 Lien entre flot maximal et coupe minimale……. 46
3.4 Recherche de flot maximal…….. 48
3.4.1 Recherche de flot maximal par chemins améliorants…… 49
3.4.1.1 Réseau résiduel…….. 49
3.4.1.2 Chemin améliorant…….. 50
3.4.1.3 Algorithme de Ford et Fulkerson……. 51
3.4.1.4 Variantes et améliorations……... 52
3.5 Coupe minimale dans un graphe non-orienté…….. 53
3.7 Conclusion……... 54

Chapitre 4 : Segmentation d'images par coupure de graphe
4.1 Introduction……. 55
4.2. Segmentation d'objets par la coupe de graphe..……. 56
4.2.1 Problème d'étiquetage ou labellisation des pixels……. 56
4.2.2 Minimisation de la fonction d'énergie…… 57
4.2.3 Formulation du graphe dans l'image…….……... 58
4.2.4 Calcul des capacités pour une configuration terminale……..…... 59
4.2.5 Calcul des capacités pour une configuration de voisinage……..…….. 60
4.3 Implantation de la technique de segmentation……..… 61
4.3.1 Apprentissage……..... 61
4.3.2 Segmentation d'image par minimisation d'énergie……... 61
4.3.3 Organigramme générale de la technique de segmentation implémentée ……... 62
4.4 Résultats expérimentaux…….. 62
4.4.1 Effets du bruit sur la segmentation ……... 65
4.4.2 Segmentation des images en couleurs ……...……... 66
4.5 Conclusion ……....……. 68

Conclusion générale ……..… 69
Bibliographie …… 71

Table des figures

1.1 Les cliques associées à divers systèmes de voisinage…….……..... 5

2.1 Graphe orienté…….... 21

2.2 Graphe orienté et réflexivité……..…..…. 21

2.3 Symétrique d'un graphe orienté …….……. 22

2.4 Graphe orienté et symétricité……..…. 22

2.5 Fermeture symétrique …….. 22

2.6 Graphe non-orienté……... 24

2.7 Opérations ensemblistes dans les graphes…… 26

2.8 Graphes complets……. 28

2.9 Graphes non-orientés et leurs dérivés…….. 31

2.10 Cliques maximales dans les graphes dérivés…… 31

2.11 Les neuf graphes minimaux non-dérivés…….…… 33

2.12 Graphes dérivés ou non……...…. 33

2.13 Graphes orientés et leurs dérivés…….. 34

2.14 Dérivation de graphes pondérés……... 37

2.15 Image 2D de taille 3 × 3 (3 pixels par 3 pixels)……..…... 40

2.16 Equivalence entre image et graphe …….. 41

3.1 Réseau de transport et flots …….……. 45

3.2 Equivalence entre coupes sur graphe orienté ou non …….. 47

3.3 Graphe résiduel…… 49

3.4 Flot complet non maximal …….. 50

3.5 Déroulement de l'algorithme de Ford et Fulkerson ……..…….. 52

3.6 Equivalence entre coupe minimales sur graphe orienté ou non……... 54

4.1 Illustration d'une coupe de graphe sur une image de dimension 3x4…….. 59

4.2 Répartition des niveaux de gris des pixels suivant la classe objet et la classe arrière plan.. 60

4.3 Organigramme générale de la technique de segmentation implémentée ……..... 63

4.4 Segmentation d'une image IRM cérébrale……... 64

4.5 Segmentation d'une image IRM de l'abdomen…… 64

4.6 Segmentation des images bruités : a) IRM cérébrale ; b) IRM de l'abdomen……. 65

4.7 Segmentation d'une image IRM cérébrale en couleur…….. 66

4.8 Segmentation de la région en rouge (tumeur) d'une image IRM cérébrale…….. 67

Introduction Générale

Introduction générale

La recherche en imagerie médicale est une des disciplines les plus actives du traitement d'images. La segmentation et l'analyse d'images dans un contexte clinique reste un problème majeur de l'imagerie médicale. La multiplicité des modalités d'imagerie, ainsi que les fortes variabilités des structures et pathologies à analyser rendent cette tâche fastidieuse. Dans la plupart des cas, la supervision de spécialistes, tels que des radiologistes, est nécessaire pour valider ou interpréter les résultats obtenus par analyse d'images. L'importante quantité de données, ainsi que les nombreuses applications liées à l'imagerie médicale, nécessitent des outils logiciels de très haut niveau combinant des interfaces graphiques complexe avec des algorithmes interactifs rapides.

Bien que certains outils mathématiques nécessaires, tels que la topologie ou les probabilités, apparaissent dès le XVIII^{ème} siècle, la recherche en traitement et analyse d'images ne s'est développée qu'à partir des années 1960 avec l'essor de l'informatique.

Les champs de Markov sont utilisés depuis maintenant une trentaine d'années en traitement d'images et font désormais partie des techniques de base de cette discipline. Nés à l'origine dans le cadre de la physique statistique pour étudier les phénomènes de transition de phase, ils sont rapidement appliqués aux réseaux bidimensionnels que constituent les images. Les premiers modèles sont restreints par des contraintes de causalité [ABEND65] [KANAL80] et donc limités jusqu'à l'article fondateur de Geman et Geman en 84 [GEMAN84a] qui ouvre réellement leur utilisation en traitement d'images. Cette utilisation ira croissant avec les années et donnera lieu à de nombreuses recherches sur les problèmes que suscite leur utilisation (calcul des paramètres des modèles, techniques de simulation et de recherche de solutions rapides, etc.).

Les graphes font également partie des outils mathématiques couramment utilisés en traitement et analyse d'images pour, notamment, la segmentation. Bien que l'on considère souvent la théorie des graphes comme remontant également aux années 1960 avec les bases posées par Claude Barge [BERGE62], leur première apparition est due à Leonhard Euler en 1736.

Depuis les travaux de Leonhard Euler, le champ d'application des graphes s'est grandement répandu dans de nombreux domaines. Les graphes sont désormais susceptibles de représenter divers réseaux (routiers, ferrés, de canalisations, électriques, informatiques,...) et d'être utilisés dans la résolution des problèmes qui y sont liés (recherche du débit maximal, du plus court chemin, de la tournée d'un livreur la plus courte, du goulot d'étranglement du réseau,...).

Comme déjà annoncés, les graphes sont également très utilisés en traitement et analyse d'images puisque, une image peut être considérée comme un graphe. De nombreux algorithmes sur les graphes ont donc pu être appliqués dans ce domaine. La segmentation d'images étant un problème récurrent en traitement et analyse d'images, il n'est donc pas étonnant que la segmentation de graphes le soit également.

La segmentation interactive d'images est devenue ces deux dernières décennies un axe de recherche très actif et en plein essor de développement. On recourt de plus en plus fréquemment à ce type d'approche pour contourner le problème inhérent à la segmentation automatique qui donne rarement des résultats parfaits.

De nouvelles approches apparaissaient inévitablement qui, au lieu d'effectuer une segmentation globale et non supervisée sur l'image, se sont appropriées du créneau pour réaliser une segmentation avec une connaissance a priori, puisque dans la plupart des cas de traitements, l'objet à segmenter est en général connu à l'avance. Ces résultats peuvent être obtenus à partir d'une plate forme qui utilise des méthodes de coupe de graphes qui sépare l'objet d'intérêt du fond de l'image.

Une méthode de minimisation par coupes minimales de graphes fut utilisée pour la première fois en vision par ordinateur en 1989 par D. M. Greig, B. T. Porteous et A. H.Seheult [GREIG89] dans le cadre d'une restauration d'image binaire en utilisant le modèle de Potts. Leur objectif était de mettre en évidence que la méthode de minimisation par recuit simulé peut donner en résultat un minimum local souvent éloigné du minimum global. Pour ce faire, ils développèrent une méthode de minimisation par coupe minimale qui donne en résultat le minimum global.

En 1998 et en 1999, près d'une décennie plus tard, Yuri Boykov, Olga Veksler et Ramin Zabih généralisèrent cette méthode au cas de n étiquettes selon certaines conditions suffisantes sur l'énergie à minimiser dans [BOYKOV98, 99 et 01].

En 2002, Vladimir Kolmogorov et Ramin Zabih énoncèrent dans [KOLMO02] une nouvelle condition sur l'énergie à minimiser, nécessaire et suffisante, permettant l'utilisation des coupes minimales comme méthode de résolution dans le cas binaire. Toutefois, cette condition peut facilement être utilisée dans le cas général de *n* étiquettes.

Il est également à noter que, dans le même article, la minimisation par coupes minimales est étendue au cas de champs aléatoires de Markov avec des fonctions sur des cliques à trois sites (et non plus deux comme on s'y restreint habituellement. Ces résultats furent repris et généralisés pour des cliques à plus de trois sites par Daniel Freedman et Petros Drineas en 2005 dans [FREED05]. Néanmoins, dans le cadre de ce mémoire, nous nous en tiendrons au cas de cliques à deux sites uniquement telle que définies dans ([FREED05], [KOLMO02]).

Ce mémoire s'organise comme suit:

Le chapitre 1 est un chapitre d'introduction aux techniques markoviennes et établit des notions de base; il donne la définition des champs markoviens et des champs de Gibbs; l'application en segmentation est également mentionnée. Ensuite l'estimateur MAP (Maximum à posteriori) est présenté.

Le chapitre 2, en premier lieu, expose la liste des définitions de base nécessaire à la manipulation de graphes, orientés et non-orientés et nous présentons des concepts plus complexes sur les graphes.

Le chapitre 3 présente un algorithme de recherche de flot maximal entre une source et un puits dans un graphe orienté appelé réseau. L'équivalence entre cette recherche et celle de coupe minimale est rappelée.

Le chapitre 4 rappelle la définition d'un problème d'étiquetage lié à un champ aléatoire de Markov et la façon dont ce dernier peut être ramené à un problème de minimisation d'énergie. Nous présentons ensuite la technique de résolution reposant sur les coupes minimales de graphes dans le cas d'un étiquetage binaire. Ensuite les résultats expérimentaux obtenus sont présentés.

CHAPITRE 1 :

Champ de Markov en traitement d'images

1.1 Introduction :

L'image numérique se présente et se manipule sous forme d'un tableau bidimensionnel (ou n-dimensionnel) d'une variable entière quantifiée. L'information véhiculée par cette représentation va en réalité bien au-delà de la seule donnée des niveaux de gris en chaque site, et la description d'une image se fait en termes de zones, contours, structures définis par les contrastes, textures, etc. qui peuvent être présents dans l'image. Le niveau de gris en un site n'est donc souvent pas significatif en lui-même, mais dans ses relations et interactions avec les pixels voisins.

Cette propriété des images, à savoir les interactions locales entre niveaux de gris voisins pour définir les différentes régions de l'image, va nous permettre d'utiliser un formalisme markovien dans de nombreux traitements, qu'il s'agisse de segmentation, de restauration et plus tard d'analyse complète des images. Le principe est de définir des énergies locales entre groupes de sites reflétant les interactions entre niveaux de gris. L'énergie globale est alors reliée à la probabilité d'apparition de l'image dans le cadre des champs de Gibbs.

Dans ce chapitre, nous introduisons tout d'abord de façon intuitive la notion d'énergie locale avant de définir plus formellement un champ de Markov et d'énoncer le théorème d'équivalence entre champs de Markov et champs de Gibbs. Les modèles markoviens les plus courants sont ensuite présentés. L'utilisation des MRF (Marcov Random Field) en traitement d'images dans un cadre bayésien, montre la nécessité de pouvoir accéder aux configurations les plus probables d'un champ markovien.

1.2 Description et modélisation probabiliste de l'image :

1.2.1 Description de l'image :

L'image est formée d'un ensemble fini S de sites s_i correspondant aux pixels. S est donc essentiellement un réseau discret fini, partie de $\mathbb{Z}^d$, si on note d la dimension de l'espace (2 le plus classiquement, 3 pour les volumes, etc.). A chaque site est associé un descripteur, représentant l'état du site et qui peut être son niveau de gris, une étiquette (label), ou une information plus complexe, et prenant ses valeurs dans E.

La notion d'interactions locales nécessite de structurer les relations spatiales entre les différents sites du réseau. Pour ce faire, on munit S d'un système de voisinage $\mathcal{V}$ défini de la façon suivante:

$$\mathcal{V}_s = \{t\} \text{ tels que } \begin{cases} s \notin \mathcal{V}_s \\ t \in \mathcal{V}_s \Rightarrow s \in \mathcal{V}_t \end{cases} \quad (1.1)$$

A partir d'un système de voisinage, un système de cliques peut être déduit : une clique est soit un singleton de *S*, soit un ensemble de sites tous voisins les uns des autres. En fonction du système de voisinage utilisé, le système de cliques sera différent et fera intervenir plus ou moins de sites comme illustré sur la figure 1.1. On notera *C* l'ensemble des cliques relatif à $\mathcal{V}$, et C_k l'ensemble des cliques de cardinal *k*.

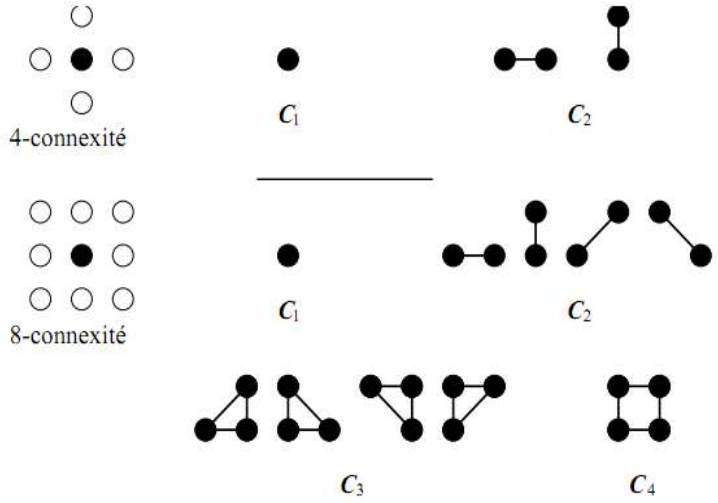


Figure 1.1 – Les cliques associées à divers systèmes de voisinage en dimension d = 2

Les interactions locales entre niveaux de gris (ou descripteurs) de sites voisins peuvent alors s'exprimer comme un potentiel de clique. Soit *c* une clique, on lui associe le potentiel U_c dont la valeur dépend des niveaux de gris (ou descripteurs) des pixels constituant la clique. En poursuivant ce raisonnement, on peut définir l'énergie globale de l'image comme la somme des potentiels de toutes les cliques :

$$U = \sum_{c \in C} U_c \quad (1.2)$$

et l'énergie locale en un site comme la somme des potentiels de toutes les cliques auxquelles il appartient :

$$U_s = \sum_{c \in C / s \in c} U_c \quad (1.3)$$

Nous avons jusqu'ici considéré le cas d'une image pour illustrer les notions de voisinage, de clique et de potentiel, mais le formalisme markovien se définit très généralement sur tout graphe. Soit un ensemble de sites *S* dénombrables (sommets du graphe), et une relation de voisinage, les cliques sont alors définies comme les sous-graphes complets du graphe. C'est l'utilisation de graphes plus généraux que ceux définis sur la grille de l'image qui permettent des traitements de plus haut niveau.

1.2.2 Modélisation probabiliste de l'image :

La définition des champs de Markov qui sera donnée dans la section suivante nécessite une modélisation probabiliste de l'image. Ainsi, l'image dont nous disposons va être considérée comme une réalisation d'un champ aléatoire. Soit *s* un site de l'image, on peut en effet lui associer une variable aléatoire X_s prenant ses valeurs dans *E*. Le niveau de gris x_s en *s* n'est ainsi qu'une réalisation de la variable aléatoire X_s. On définit alors le champ aléatoire $X = (X_s, X_t, ..)$ prenant ses valeurs dans $\Omega = E^{|S|}$ On trouvera aussi le terme de processus aléatoire pour *X*; en toute rigueur, "processus" devrait être réservé au cas d'un ensemble d'indexation continu, et champ au cas discret.

Dans ce cadre probabiliste, l'image considérée est simplement une réalisation *x* du champ. La probabilité globale de *x*, $P(X = x)$, permet d'accéder en quelque sorte à la vraisemblance de l'image, et les probabilités conditionnelles locales d'une valeur en un site permettent de mesurer le lien statistique entre un niveau de gris et le reste de l'image. L'hypothèse markovienne permet d'évaluer ces quantités.

Notons que nous nous plaçons dans le cas où *E*, l'espace de valeurs des descripteurs, est quantifie, ce qui nous permet de manipuler des probabilités. Dans le cas ou cet espace est continu, il faut remplacer *P* par une densité de probabilité, mais dans ce cas le théorème que nous allons voir ci-dessous n'est plus valable.

1.3 Champs de Markov - Champs de Gibbs :

1.3.1 Définition d'un champ de Markov :

Considérons x_s la valeur du descripteur prise au site s et $x^s = (x_t)_{t \neq s}$ la configuration de l'image excepté le site *s*. La définition d'un champ de Markov est alors la suivante :

X est un champ de Markov si la probabilité conditionnelle locale en un site n'est fonction que de la configuration du voisinage du site considéré.

Ce qui s'exprime de façon formelle par :

$$P(X_i = x_i / x^i) = P(X_i = x_i / x_j, j \in \mathcal{V}_i) \qquad (1.4)$$

Ainsi, le niveau de gris en un site ne dépend que des niveaux de gris des pixels voisins de ce site. Cette hypothèse markovienne se justifie bien dans le cas de la plupart des images naturelles constituées de zones homogènes ou texturées ainsi que pour une large gamme d'images de synthèse. Plus généralement, une connaissance locale de l'image suffit souvent à réaliser son interprétation partielle et donc cette hypothèse markovienne sera souvent justifiée sur des graphes plus globaux que le graphe des pixels.

Notons qu'en l'absence de contrainte sur le système de voisinage, tous les champs aléatoires peuvent être considérés comme markoviens à condition de prendre un voisinage suffisamment grand. L'intérêt de cette modélisation réside bien sûr dans le cas où la propriété markovienne est vérifiée pour des voisinages restreints permettant des calculs rapides.

1.3.2 Equivalence champs de Markov - champs de Gibbs :

La modélisation markovienne prend toute sa puissance grâce au théorème que nous allons voir maintenant. En effet, celui-ci permettra d'accéder aux expressions des probabilités conditionnelles locales. Il nous faut au préalable définir un certain nombre de notions relatives aux mesures et champs de Gibbs.

Définition d'une mesure de Gibbs: La mesure de Gibbs de fonction d'énergie (ou d'Hamiltonien) $U : \Omega \rightarrow \mathbb{R}$ est la probabilité *P* définie sur Ω par :

$$P(X = x) = \frac{1}{Z}\exp(-U(x)) \qquad (1.5)$$

avec

$$U(x) = \sum_{c \in C} U_c(x) \qquad (1.6)$$

où *C* est le système de cliques associé au système de voisinage $\mathcal{V}$ de U^2. $Z = \sum_{x \in \Omega} \exp(-U(x))$ est une constante de normalisation appelée fonction de partition de Gibbs. En pratique, il est quasi impossible de calculer cette constante à cause du très grand nombre de configurations possibles. Ne serait-ce que dans le cas d'une image binaire (*Card*(*E*) = 2) et de taille = 512 x 512, on a 2^{262144} configurations possibles.

La notation couramment utilisée pour *U(x)* est abusive car $U_c(x)$ ne dépend pas de l'ensemble de la configuration *x* mais seulement de *x* restreinte à la clique *c* ($U_c(x) = U_c(x_j, j \in c)$).

Nous pouvons maintenant définir le champ de Gibbs de potentiel associé au système de voisinage $\mathcal{V}$: c'est le champ aléatoire *X* dont la probabilité est une mesure de Gibbs associée au système de voisinage $\mathcal{V}$, ce qui implique:

$$P(X = x) = \frac{1}{Z}\exp\big(-U(x)\big) = \frac{1}{Z}\exp(-\sum_{c\in C} U_c(x)) \qquad (1.7)$$

L'énergie globale d'un champ de Gibbs possède donc la propriété de se décomposer sous forme d'une somme d'énergies locales, qui comme on le verra par la suite permettront d'accéder aux probabilités conditionnelles locales. Notons ici que plus une configuration d'un champ de Gibbs a une énergie faible, plus elle est probable.

Le théorème de Hammersley-Clifford [BESAG74] établit alors le résultat fondamental suivant sous les hypothèses :

- *S* fini ou dénombrable,
- système de voisinage $\mathcal{V}$ borné,
- espace des états *E* discret

X est un champ de Markov relativement à $\mathcal{V}$ et $P(X = x) > 0 \quad \forall x \in \Omega \Leftrightarrow$ *X* est un champ de Gibbs de potentiel associé à $\mathcal{V}$

Par exemple, si nous considérons un champ de Markov de voisinage 4-connexe, nous pouvons écrire l'énergie de la configuration *x* sous la forme :

$$U(x) = \sum_{c=(i)\in C_1} U_c(x_i) + \sum_{c=(i,j)\in C_2} U_c(x_i, x_j) \qquad (1.8)$$

Notons que rien n'interdit la non-stationnarité du champ, c'est à dire la variation des potentiels U_c en fonction de la localisation de la clique c dans l'image [DESCOM93]. D'autre part, rien n'impose la symétrie des potentiels et on peut avoir $U_{c=(r,s)}(0,1) \neq U_c(1,0)$.

Le théorème de Hammersley-Clifford, et la forme bien spécifique de probabilité de *X* qui en résulte, va permettre de lier les probabilités globales et locales comme nous allons le voir maintenant. En effet si nous cherchons à écrire la probabilité conditionnelle locale $P(x_s / X^S = x^S)$, nous avons grâce au résultat précédent :

$$P(X_s = x_s/X^s = x^s) = \frac{P(X = x)}{P(X^s = x^s)} = \frac{\exp(-U(x_s, x^s))}{\sum_{x_s \in E} \exp(-U(x_s, x^s))} \quad (1.9)$$

Définissons l'énergie locale U_s par:

$$U_s(x_s / x_t, t \in V_s) = \sum_{c \in C / s \in c} U_c(x_s, x_t, t \in V_s)$$
$$= \sum_{c \in C / s \in c} U_c(x_s, V_s) \quad (1.10)$$

en notant $V_s = (x_t, t \in \mathcal{V}_s)$. Cette énergie locale ne fait donc intervenir que les voisins de *i*. On peut alors écrire l'énergie globale *U(x)* sous la forme :

$$U(x) = \sum_{c \in C / s \notin c} U_c(x) + \sum_{c \in C / s \in c} U_c(x)$$
$$= \sum_{c \in C / s \notin c} U_c(x) + U_s(x_s / V_s) \quad (1.11)$$

En simplifiant l'expression de la probabilité conditionnelle locale en supprimant les termes communs qui font intervenir les cliques ne contenant pas le site i au numérateur et au dénominateur, on a:

$$P(X_s = x_s / X^s = x^s) = \frac{\exp(-\sum_{c \in C / s \notin c} U_c(x) - U_s(x_s/V_s))}{\sum_{x_s \in \Lambda} \exp(\sum_{c \in C / s \notin c} U_c(x) - U_s(x_s/V_s))}$$
$$= \frac{\exp(-U_s(x_s/V_s))}{\sum_{x_s \in \Lambda} \exp(-U_s(x_s/V_s))} \quad (1.12)$$

L'expression obtenue, qui ne fait intervenir que les potentiels des cliques contenant le site *s* (ce qui nous permet de retrouver au passage l'hypothèse markovienne), est très importante. En effet, autant il n'est pas possible partant d'une configuration *x* d'accéder à sa probabilité à cause de la constante de normalisation, autant il est possible de calculer en chaque site la probabilité conditionnelle locale. Cette expression sera à la base de tous les algorithmes de simulation de champs markoviens que nous verrons dans la section suivante.

Remarque :

Le théorème de Hammersley-Clifford n'est valable que lorsqu'aucune configuration n'est interdite (condition $P(X = x) > 0 \ \forall x$).

1.4 Quelques MRF fondamentaux :

Nous présentons ici quelques uns des champs de Markov les plus utilisés. Comme indiqué précédemment, ces champs sont définis par leur voisinage et leurs fonctions de potentiel. Ils sont illustrés par le tirage de réalisations selon l'échantillonneur de Gibbs.

Modèle d'Ising :

Ce modèle est le plus ancien (1925) et a été développé lors de l'étude du ferromagnétisme en physique statistique. L'espace des descripteurs est celui des états des spins, i.e $E = \{-1,1\}$ (espace binaire), et le voisinage est constitue par les 4 ou 8 plus proches voisins dans un espace bidimensionnel. Les potentiels sont des potentiels en tout ou rien :

$$U_{c=(s,t)}(x_s, x_t) = -\beta \quad si\ x_s = x_t$$
$$= +\beta \quad si\ x_s \neq x_t \qquad (1.13)$$

Ce qui s'écrit également :

$$U_{c=(s,t)}(x_s, x_t) = -\beta x_s x_t \qquad (1.14)$$

Les potentiels des cliques d'ordre 1 (clique constitue par un seul spin) sont de la forme $-Bx_s$. L'énergie totale s'écrit :

$$U(x) = -\sum_{c=(s,t)\in C} \beta x_s x_t - \sum_{s\in S} B\, x_s \qquad (1.15)$$

β est la constante de couplage entre sites voisins et B représente un champ magnétique externe. Lorsque β est positif, les configurations les plus probables (i.e d'énergies plus faibles) sont celles pour lesquelles les spins sont de même signes (ferromagnétisme), alors que dans le cas de β négatif, au contraire, on favorisera l'alternance de spins de signes opposés (antiferromagnétisme). La valeur (signe et valeur absolue) de β conditionne donc la régularité du modèle d'Ising. Quant au champ magnétique externe relatif au potentiel d'ordre 1, il favorise a priori par son signe un spin ou un autre.

Modèle de Potts :

Il s'agit d'une extension du modèle d'Ising [WU82] pour un espace m-aire, i.e :
$E = \{0, m-1\}$. Il peut s'agir de plusieurs niveaux de gris, mais plus souvent pour ce modèle, d'étiquettes (labels) pouvant représenter une classification de l'image (par exemple les classes eau,

foret, champ, ville). Le voisinage considéré est 4 ou 8-connexe et les potentiels sont comme précédemment en tout ou rien mais définis seulement pour les cliques d'ordre 2:

$$U_{c=(s,t)}(x_s, x_t) = -\beta \quad si\ x_s = x_t$$
$$= +\beta \quad si\ x_s \neq x_t \qquad (1.16)$$

Lorsque β est positif, les configurations les plus probables correspondent à des sites voisins de même niveau de gris ou descripteur, ce qui donne des réalisations constituées par des larges zones homogènes. La taille de ces régions est gouvernée par la valeur de β.

Il est possible de définir des modèles utilisant des pondérations β différentes en fonction des directions des cliques, et de privilégier ainsi certaines directions.

Ce modèle permet également de prendre en compte différentes relations entre les régions (i.e entre différentes valeurs des descripteurs). On peut par exemple définir des pondérations $\beta\ (e_s, e_t)$ pour $e_i, e_j \in E$. Dans notre exemple de classification en 4 étiquettes eau, foret, champ, ville, une configuration des sites avec les étiquettes champ/foret peut être supposée plus probable qu'une configuration ville/foret, d'ou des valeurs β (champ, foret) et β (ville, foret) différentes.

Modèle markovien gaussien :

Ce modèle est réserve aux images en niveaux de gris $E = \{0, \ldots, 255\}$ et ne convient pas bien aux images d'étiquettes. Le voisinage est 4 ou 8-connexe et l'énergie est de la forme:

$$U(x) = \beta \sum_{c=(s,t)} (x_s - x_t)^2 + \alpha \sum_{s \in S} (x_s - \mu_s)^2 \qquad (1.17)$$

Le premier terme correspondant aux cliques d'ordre 2 est un terme de régularisation, qui favorise les faibles différences de niveaux de gris entre sites voisins pour $\beta > 0$. Le second terme peut correspondre à un terme d'attache aux données dans le cas ou on possède une image de données extérieures. Le rapport $\frac{\alpha}{\beta}$ pondère les influences respectives de l'attache aux données et de la régularisation, et les valeurs absolues des paramètres caractérisent le caractère plus ou moins "piqué" ou équiréparti au contraire de la distribution.

Le modèle gaussien favorise des niveaux de gris proches pour des pixels voisins dans tous les cas. Or si on considère une image naturelle cet aspect est néfaste à proximité des contours car il favorisera la présence d'un dégradé. Aussi, de nombreuses fonctions φ ont été proposées pour modéliser les potentiels des cliques d'ordre 2 : $U_{c=(s,t)} = \varphi(x_s - x_t)$. En particulier, la fonction suivante permet de respecter les contours de l'image [Geman85]:

$$\varphi(x) = \frac{1}{1 + (\frac{x}{\delta})^2} \qquad (1.18)$$

et est donc très utilisée en restauration. Ces modèles permettent de synthétiser des textures très variées.

1.5 Exemple d'application : la segmentation :

Pour cette application, on peut modéliser le problème dans un cadre bayésien de la façon suivante. Nous disposons d'une certaine donnée (image) que nous noterons *y* et que nous pouvons considérer comme une réalisation d'un champ aléatoire *Y*. Nous cherchons une réalisation *x* de l'image segmentée, que nous pouvons modéliser comme un champ de Markov *X*. *X* est le champ des étiquettes (labels). Les espaces de configurations ne sont donc pas nécessairement les mêmes pour *X* et *Y*. Ces deux champs sont liés par le processus d'acquisition de l'image, qui conduit du champ idéal *X* , le processus image original que nous cherchons, au champ bruité *Y* que nous observons. La segmentation a pour objectif d'inverser le processus et donc de remonter à une réalisation de *X* a partir de l'observation des données bruitées *y*. On parle dans ce contexte de champ de Markov "caché" pour *X*, ou de données incomplètes puisque *y* n'est pas une réalisation de *X*.

On peut par exemple utiliser le critère du maximum a posteriori et rechercher la configuration $\hat{x}$ maximisant la probabilité de *X* conditionnellement à la donnée *y* i.e *P*(*X*= *x* / *Y*= *y*). Or la régle de Bayes permet d'écrire:

$$P(X = x/Y = y) = \frac{P(Y = y/X = x)P(X = x)}{P(Y = y)} \qquad (1.19)$$

Expression dans laquelle il s'agit alors d'analyser chacun des termes *P*(*Y* = *y* | *X* = *x*) et *P*(*X* = *x*), sachant que *P(Y)* est une constante (indépendante de la réalisation *x*).

Le premier terme *P*(*Y*= *y*| *X*= *x*) décrit justement le processus d'observation et d'acquisition des données. L'hypothèse la plus courante (dont la validité reste à justifier) consiste à supposer l'indépendance conditionnelle des pixels (bruit non corrélé par exemple) :

$$P(Y = y \,/\, X = x) = \prod_s P(Y_s = y_s \,/\, X_s = x_s) \qquad (1.20)$$

Cette écriture n'est plus valable lorsqu'il y a une convolution par la fonction de transfert du système d'acquisition, mais on peut montrer que le champ a posteriori reste markovien.

Par ailleurs, on fait sur le champ *X* recherché une hypothèse markovienne selon un voisinage $\mathcal{V}$ et un modèle donné dépendant de l'application. On peut alors écrire :

$$P(X = x) = \frac{\exp(-U(x))}{Z} \tag{1.21}$$

Si on revient maintenant à la distribution a posteriori, celle-ci s'exprime par:

$$\begin{aligned} P\,(X\ =\ x/Y\ =\ y)\ &\propto\ P\,(Y\ /X\,)P\,(X) \\ &\propto\ e^{lnP\,(\,Y/X\,)-U(x)} \\ &\propto\ e^{-U(x/y)} \end{aligned} \tag{1.22}$$

Avec

$$U(x/y) = \sum_{s\in S} -\ln P(y_s/x_s) + \sum_{c\in C} U_c(x) \tag{1.23}$$

Par conséquent, sous les hypothèses précédentes, on constate que la distribution a posteriori est une distribution de Gibbs et que donc le champ *X* conditionnellement à *y* est également un champ de Markov (théorème de Hammersley-Clifford). Ainsi, il est possible de simuler des réalisations de ce champ à l'aide de l'échantillonneur de Gibbs ou de l'algorithme de Métropolis. Mais la configuration *x* qui nous intéresse est celle maximisant la probabilité a posteriori, donc la réalisation la plus probable du champ de Gibbs, ou encore celle qui minimise l'énergie *U* (*x* / *y*).

Dans ce contexte, le champ markovien *X* est défini sur un autre espace de configurations que *Y* car seulement quelques étiquettes sont considérées : $E = \{0, \ldots, m-1\}$. Dans ce cas le processus de passage de *X* (champ des labels) à *Y* ne décrit pas tant le processus d'acquisition que l'apparence des classes dans l'image. Le terme $P(Y = y / X = x)$ traduit donc la probabilité de réalisation d'une configuration donnée connaissant son étiquetage (i.e connaissant la classe de chaque pixel). En supposant l'indépendance des sites les uns par rapport aux autres, et en supposant que le niveau de gris y_s en un site *s* ne dépend que de l'étiquette x_s en ce site, on a:

$$P(Y = y\ /\ X = x)\ = \prod_s P(y_s/x_s) \tag{1.24}$$

Les valeurs des probabilités conditionnelles sont données par l'histogramme conditionnel des niveaux de gris pour une classe donnée. Par exemple, si on suppose que chaque classe *i* a une distribution gaussienne de moyenne μ_i et d'écart-type σ_i, on a :

$$P\ (y_s\ /\ x_s = i) = -\frac{1}{\sqrt{2\pi\sigma_i}}\ exp(-\frac{(y_s - \mu_i)^2}{2\sigma_i^2}) \tag{1.25}$$

Si comme précédemment on fait une hypothèse markovienne sur *X* et qu'on se limite aux cliques d'ordre 2, on a :

$$P(X = x) = \frac{1}{Z} exp(-\beta \sum_{(s,t)\in C_2} \varphi(x_s, x_t)) \quad (1.26)$$

D'où l'énergie a posteriori:

$$U(x/y) = \sum_s \frac{(y_s - \mu_{x_s})^2}{2\sigma_{x_s}^2} + \log\sqrt{2\pi\sigma_{x_s}} + \beta \sum_{(s,t)\in C_2} \varphi(x_s, x_t) \quad (1.30)$$

Le champ des labels conditionnellement à ***y*** est markovien et d'énergie de Gibbs *U*(*x / y*)**.** Là encore, le terme d'ordre 1 exprime le respect des données (le niveau de gris doit correspondre à la classe), et le terme d'ordre 2 la contrainte de régularisation introduite. On choisit souvent un modèle de Potts pour *X*, ce qui donne une image segmentée avec de larges zones homogènes.

1.6 Estimateur dans un cadre markovien :

1.6.1 Introduction :

Nous avons vu auparavant comment il était possible d'utiliser le formalisme markovien à des fins de segmentation. On se situe alors dans le cadre de données incomplètes (on parle aussi de champs de Markov cachés) car la réalisation dont on dispose est une réalisation bruitée (ou plus généralement vue a travers le système d'acquisition) du champ de Markov originel. En notant *Y* le champ dont on observe une réalisation, et *X* le champ initial, l'objectif est alors d'obtenir la "meilleure" réalisation $\hat{x}$ de *X* connaissant l'observation *y*, autrement dit, reconstruire *x* de manière optimale vis à vis d'un certain critère. Nous nous étions intéressés à la réalisation maximisant la probabilité a posteriori *P*(*X = x / Y = y*), et nous pouvons utiliser un algorithme permettant d'obtenir cette réalisation: le recuit simulé [GEMAN84] et [KIRKPA82]. En réalité d'autres choix sont possibles, auxquels correspondent d'autres méthodes de résolution, et que nous allons aborder dans la partie suivante.

1.6.2 Modélisation bayésienne et fonction de coût :

Si nous reprenons rapidement le raisonnement effectue auparavant, on peut écrire, en appliquant la règle de Bayes (équation 1.19), on montre alors que sous certaines hypothèses (indépendance des sites dans la probabilité conditionnelle *P (Y / X)* et hypothèse markovienne pour le champ *X*), la distribution a posteriori est une distribution de Gibbs et donc que le champ *X* conditionnellement à

la donnée *y* est markovien. Cette propriété n'est pas nécessaire pour les notions suivantes, mais elle sera primordiale pour les algorithmes de résolution.

Le problème est alors de déterminer une estimation $\hat{x}$ optimisant un certain critère, où $\hat{x}$ est une fonction déterministe φ de la donnée y :

$$\hat{x} = \varphi(y) \text{ avec } \varphi : \Omega \longrightarrow \Omega \tag{1.31}$$

L'estimation bayésienne procède alors comme suit. On se donne une fonction de coût, *L* définie de $\Omega \times \Omega$ dans $\mathbb{R}^+$, qui représente le coût de remplacer *x* par $\varphi(y)$, et qui possède les propriétés suivantes :

$$\forall x, x' \in \Omega \times \Omega : \tag{1.32}$$
$$\bullet\ L(x, x') \geq 0$$
$$\bullet\ L(x, x') = 0 \iff x = x'$$

L'estimateur optimal, i.e la fonction φ optimale est alors la fonction minimisant l'espérance (notée *E* [ ]) du coût, c'est à dire :

$$E\left[L(X, \varphi(y))\ /Y = y\right] = \sum_{x \in \Omega} L(x, \varphi(y)) P(x\ /\ y) \tag{1.33}$$

La fonction φ^{opt} minimise donc l'erreur moyenne conditionnellement à *y*, et l'estimateur "optimal" est $\hat{x} = \varphi^{opt}(y)$.

Suivant les fonctions de coût envisagées, on obtient différents estimateurs et différentes méthodes de résolution associées.

a) L'estimateur MAP :

Considérons la fonction de coût suivante :

$$L(x, x') = 1 \quad si\ x \neq x'$$
$$L(x, x') = 0 \quad sinon \tag{1.34}$$

Cette fonction consiste donc à pénaliser toute différence entre deux configurations, et ce, quelque soit le nombre de sites en lesquels elles diffèrent. Nous pouvons alors écrire :

$$E\,[L(X,\varphi(y))\,/y] = \sum_{x\in\Omega} L(x,\varphi(y))P(x\,/\,y)$$
$$= \sum_{x\neq\varphi(y)} P(x/y)$$
$$= 1 - \sum_{x=\varphi(y)} P(x/y)$$
$$= 1 - P(X=\varphi(y)/y) \qquad (1.35)$$

Par conséquent, la fonction φ^{opt} minimisant l'espérance pour cette fonction de coût, est celle qui maximise la probabilité a posteriori :

$$\hat{x} = \varphi^{opt}(y) = Argmax_{\varphi}[P(X=\varphi(y)/y] \qquad (1.36)$$

Il nous faut donc trouver la réalisation $\hat{x}$, fonction de *y*, maximisant la probabilité a posteriori *P*(*X / y*). On parle de l'estimateur MAP (maximum a posteriori) ou de maximum de vraisemblance a posteriori.

On retrouve donc avec cette fonction de coût en tout ou rien, la démarche intuitive que nous avions présentée, à savoir chercher la configuration maximisant la probabilité conditionnellement à la donnée disponible.

b) Solution algorithmique de l'estimateur MAP dans le cas markovien :

Le MAP nécessite l'évaluation de :

$$\hat{x} = Argmax_x\, P(X=x/y) \qquad (1.37)$$

Or la taille de l'espace des configurations Ω ne permet pas un calcul direct des quantités *P*(*x/y*) et *P*(x_S/*y*). Aussi réalise-t-on en pratique des approximations de type Monte-Carlo. En effet, supposons que l'on soit capable de tirer des réalisations de *X* selon sa loi conditionnelle à *y*, et notons les *x(1),…, x(N)*. Le tirage des réalisations ne pose quant à lui pas de problème particulier car on peut tirer des réalisations d'un champ de Gibbs avec l'échantillonneur de Gibbs et l'algorithme de Métropolies. Or sous les hypothèses rappelées en début de la section 1, la probabilité a posteriori *P(X / y)* est une distribution de Gibbs.

Dans ce cas, la connaissance de *N* échantillons tirés selon la loi a posteriori n'est pas suffisante pour obtenir une réalisation correspondant au minimum global de l'énergie. Il est possible

d'obtenir l'estimateur MAP par un recuit simulé utilisant la distribution à posteriori avec paramètre de température.

L'idée d'intégrer un paramètre de température et de simuler un recuit a été initialement proposée par Kirkpatrick [KIRKPA82] et repris par Geman et Geman [GEMAN84]. Malheureusement, cet algorithme est très lourd en temps de calcul puisqu'il demande la génération d'un grand nombre de configurations au fur et à mesure que la température décroît.

Des algorithmes sous-optimaux sont donc souvent utilisés en pratique. Besag [Besag86] a ainsi proposé un autre algorithme, beaucoup plus rapide, mais pour lequel nous n'avons pas de preuve de convergence vers un minimum global. Il s'agit de l'ICM (Iterated Conditional Mode), cet algorithme, comme les échantillonneurs, est un algorithme itératif modifiant à chaque étape les valeurs x_s de l'ensemble des sites de l'image. Mais à la différence de ces algorithmes qui étaient stochastiques, la modification d'une valeur se fait ici de façon déterministe.

Une autre méthode de minimisation par coupes minimales de graphes [GREIG89] dans le cadre d'une restauration d'image binaire. Pour ce faire, ils développèrent une méthode de minimisation par coupe de graphe minimale qui donne en résultat le minimum global.

1.7 Conclusion :

Nous avons abordé dans ce chapitre les notions fondamentales du formalisme markovien. Nous avons vu comment se définissait un champ de Markov et comment l'équivalence champ de Markov-champ de Gibbs permettait de définir sa loi de probabilité en termes d'énergie. Nous avons présenté l'estimateur bayésien MAP, et l'algorithme qui lui est associée dans un cadre markovien.

Le chapitre suivant dresse la liste des définitions et terminologies de base nécessaire à la manipulation de graphes, orientés et non-orientés. Nous présentons ensuite des concepts plus complexes sur les graphes qui s'avèrent utiles pour la bonne compréhension de la suite de ce mémoire.

CHAPITRE 2 :

Généralités sur les graphes

2.1 Introduction :

Ce deuxième chapitre présente les bases nécessaires à la bonne compréhension de la suite de ce mémoire.

Nous commençons donc par quelques rappels de définitions et de notations sur les ensembles qui nous permettront ensuite de définir les graphes orientés et non-orientés. Nous présentons ensuite les notions et propriétés liées aux graphes dont nous aurons besoin par la suite. Enfin, nous mettons en évidence le lien fort existant entre image et graphe.

2.2 Les ensembles :

Un ensemble est une collection d'objets distinguables, appelés éléments ou membres. Si un objet *x* est un élément d'un ensemble *E*, alors on a $x \in E$ (lire " *x* appartient à *E* "). Si *x* n'est pas un élément de *E*, alors on a $x \notin E$ (lire " *x* n'appartient pas à *E* "). Un ensemble ne peut pas contenir un même objet plus d'une fois et ses éléments ne sont pas ordonnés.
Un ensemble est dit fini si le nombre d'éléments qui le composent est un entier naturel, sinon, il est infini.

Notations et définitions :

Un ensemble fini peut être décrit en énumérant explicitement chacun de ses éléments dans une liste entourée d'accolades (exemple : $E = \{b, a, c\}$. L'ensemble vide est représenté par $\emptyset$, c'est l'ensemble ne contenant aucun élément.

Soient A et B deux ensembles finis d'éléments. Ci-dessous se trouvent quelques notations et définitions usuelles liées aux ensembles :

– Deux ensembles sont égaux, et l'on note $A = B$, s'ils contiennent les mêmes éléments.

– Le cardinal (ou taille) d'un ensemble, noté $|A|$, est le nombre d'éléments de l'ensemble. Bien sûr, $|\emptyset| = 0$.

– L'ensemble *A* est inclus dans l'ensemble *B*, noté $A \subseteq B$, si tout élément de *A* se trouve également dans *B*, on dit alors que *A* est un sous-ensemble de *B* ou encore une partie de *B*.

– L'ensemble *A* est inclus strictement dans l'ensemble *B*, noté $A \subset B, si\ A \subseteq B\ et\ A \neq B$, on dit alors que *A* est un sous-ensemble strict de *B*.

– L'union des ensembles *A* et *B*, noté $A \cup B$, est l'ensemble des éléments qui sont dans *A* ou dans *B*.

$$A \cup B = \{x \mid x \in A\ ou\ x \in B\}$$

– L'intersection des ensembles *A* et *B*, noté $A \cap B$, est l'ensemble des éléments qui sont dans *A* et dans *B*.

$$A \cap B = \{x | x \in A \; et \; x \in B\}$$

– La différence (ensembliste) des ensembles *A* et *B*, noté $A \setminus B$ (prononcé *A "moins" B*), est l'ensemble des éléments qui sont dans *A* mais pas dans *B*.

$$A \setminus B = \{x \mid x \in A \; et \; x \notin B\}$$

– L'ensemble des parties de *A*, noté $\mathcal{P}(A)$, est l'ensemble de tous les sous-ensembles de *A*. Conséquemment, si $A \subseteq B$, alors $A \in \mathcal{P}(B)$ (il est à noter que $|\mathcal{P}(A)| = 2^{|A|}$, ce qui justifie la notation, également usuelle, $\mathcal{P}(A) = 2^A$).

– Le complémentaire de *A* dans l'ensemble *B*, noté $\bar{A}^B$ ou plus simplement *A* quand l'ensemble *B* est une référence explicite, est l'ensemble $B \setminus A$.

– Un couple de deux éléments, noté (a, b), est une paire ordonnée d'éléments. Ainsi on a $(a, b) \neq (b, a)$.

– Le produit cartésien de *A* par *B*, noté $A \times B$, est l'ensemble de tous les couples tel que le premier élément soit dans *A* et le second soit dans *B*. Il est à noter que, en règle générale, $A \times B \neq B \times A$.

$$A \times B = \{ (a, b) \mid a \in A \; et \; b \in B\}$$

2.3 Graphes orientés et non-orientés :

Nous allons voir qu'il y a plusieurs types de graphes et que chacun d'entre eux peut être défini de plusieurs façons.

2.3.1 Graphes orientés:

Un graphe orienté *G* est un couple (*S*, *A*) où *S* est un ensemble fini d'éléments, appelés sommets, et où *A* est un ensemble de couples, appelés arcs, qui est donc un sous-ensemble de $S \times S$ (i.e. une relation sur *S*).

Si (*x*, *y*) est un arc du graphe orienté *G* = (*S*, *A*), alors on dit que :

– l'arc (*x*, *y*) part du sommet *x* et arrive au sommet *y*,

– l'arc (*x*, *y*) relie le sommet *x* au sommet *y*,

– les sommets *x* et *y* sont les extrémités de l'arc (*x*, *y*),

– *x* est le prédécesseur de *x*,

– *y* est le successeur de *y*,

– *x* est le sommet de départ de l'arc (*x*, *y*),

– *y* est le sommet d'arrivée de l'arc (*x*, *y*), et

– le sommet *y* est adjacent au sommet *x* (il est à noter que la réciproque, à savoir que le sommet *x* est adjacent au sommet *y*, n'est vraie que si l'arc (*y*, *x*) est également dans *A*).

Les sommets *x* et *y* sont dits voisins s'il existe un arc (*x*, *y*) ou (*y*, *x*) dans A. Un arc reliant un sommet à lui-même est appelé boucle.
Soient $a_1 = (x_1, y_1)$ et $a_2 = (x_2, y_2)$ deux arcs du graphe $G = (S, A)$. On dit que l'arc a_2 est adjacent à l'arc a_1 si $y_1 = x_2$.

Pour tout sommet *x* d'un graphe orienté on définit le degré rentrant, noté $d^-(x)$, et le degré sortant, noté $d^+(x)$, qui sont respectivement le nombre d'arcs qui arrivent au sommet *x* et le nombre de sommets qui partent du sommet *x*. On dit d'un sommet qu'il est isolé quand ses degrés rentrant et sortant sont nuls.

Dans un graphe orienté $G = (S, A)$, on appelle chemin du sommet x_0 au sommet x_ℓ dans *G* une séquence ordonnée de sommets $\pi = \langle x_0, \ldots, x_\ell \rangle$ de sommets de *S* tels que pour tout entier $k \in [1; \ell], (x_{k-1}, x_k) \in A$. On dit que le chemin π contient les sommets $x_0, x_1, \ldots, x_\ell$ et les arcs $(x_0, x_1), (x_1, x_2), \ldots, (x_{\ell-1}, x_\ell)$. Un chemin peut également être défini par la séquence ordonnée des arcs qui le constituent : $\pi = \langle a_0, \ldots, a_{\ell-1} \rangle$. On dit que ℓ est la longueur du chemin, c'est le nombre d'arcs qu'il contient.

Un chemin du sommet x_0 au sommet x_0 de longueur nulle, autrement dit dont la séquence de sommets est $\pi = \langle x_0 \rangle$, est appelé chemin trivial.

Un chemin est dit élémentaire si les sommets qui le constituent sont tous distincts. On dit qu'un sommet *y* est accessible à partir du sommet *x* s'il existe un chemin $\pi = \langle x, \ldots, y \rangle$. On peut facilement montrer que s'il existe un chemin de *x* à *y*, alors il existe un chemin élémentaire de x à y.

Un chemin $\pi = \langle x_0, \ldots, x_\ell \rangle$ est appelé circuit si $x_0 = x_\ell$ et s'il contient au moins un arc. Si tous ses sommets sauf x_0 et x_ℓ sont distincts, on parle alors de circuit élémentaire.

Exemple 2.1:
Un exemple de graphe orienté se trouve dans la figure 2.1 (les sommets sont représentés par des cercles et les arcs par des flèches les reliant). Dans ce graphe :
– $d^-(h) = 2$; $d^+(h) = 5$;
– $\langle h, l, m, h, l, i, d, c \rangle$ est un chemin de longueur *7* ;
– $\langle h, l, m, i, d, c \rangle$ est un chemin élémentaire de longueur *5* ;
– $\langle e, f, j, e \rangle$ est un circuit élémentaire.

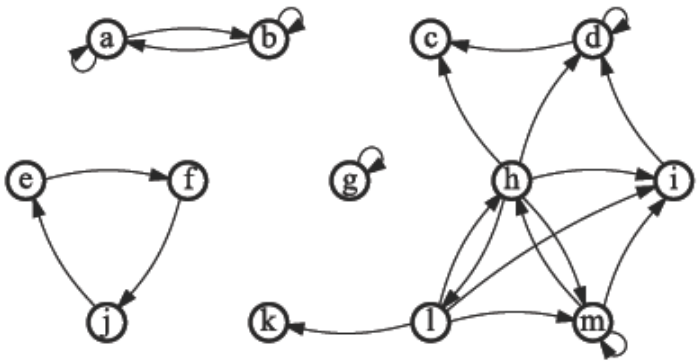


Figure 2.1 – Graphe orienté.

2.3.1.1 Graphe réflexif :

Un graphe orienté $G = (S, A)$ est dit réflexif si $\forall x \in S, (x, x) \in A$, autrement dit, si chaque sommet a une boucle.

Un graphe orienté $G = (S, A)$ est dit antiréflexif si $\forall x \in S, (x, x) \notin A$, autrement dit, si chaque sommet est sans boucle.

Des exemples de ces définitions se trouvent dans la figure 2.2.

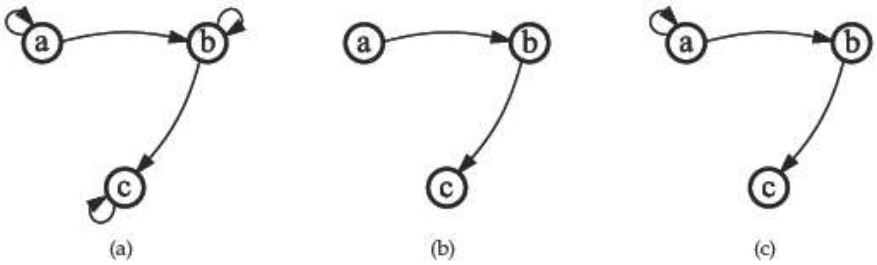


Figure 2.2 – (a) Graphe orienté réflexif ; (b) Graphe orienté antiréflexif ; (c) Graphe orienté non-réflexif et non-antiréflexif.

2.3.1.2 Graphe symétrique et fermeture symétrique :

Soit un graphe orienté $G = (S, A)$. On appelle graphe symétrique de G le graphe orienté $G^{-1} = (S, A^{-1})$ tel que $(x, y) \in A \Leftrightarrow (y, x) \in A^{-1}$.

Il s'agit d'un graphe dont l'orientation des arcs à été inversée par rapport au graphe d'origine.

Soit $a \in A$. On note a^{-1} l'arc de A^{-1} symétrique de a.

Un exemple de cette définition se trouve dans la figure 2.3.

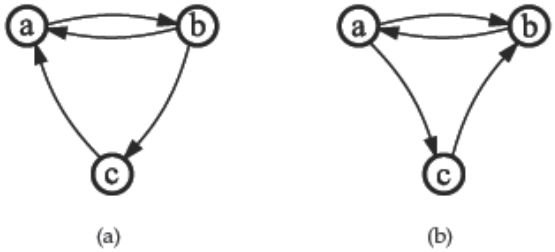


Figure 2.3 – (a) Graphe orienté G ; (b) Graphe orienté G^{-1} (symétrique de G).

Un graphe orienté $G = (S, A)$ est dit symétrique si $\forall (x, y) \in S^2, (x, y) \in A \Leftrightarrow (y, x) \in A$.
Un graphe orienté $G=(S, A)$ est dit antisymétrique si$\forall (x, y) \in S^2, (x, y) \in A \; et (y, x) \in A \Rightarrow x = y$.
Des exemples de ces définitions se trouvent dans la figure 2.4.

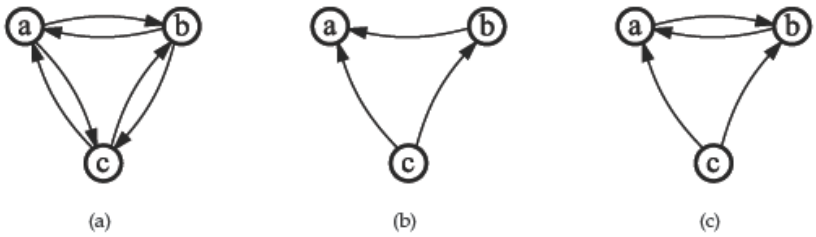


Figure 2.4 – (a) Graphe orienté symétrique ; (b) Graphe orienté antisymétrique ; (c) Graphe orienté non-antisymétrique et non-symétrique.

On appelle fermeture symétrique d'un graphe orienté $G = (S, A)$ le graphe $G'=(S, A^*)$ avec $A^* = A \cup A^{-1}$. Bien sûr le résultat d'une fermeture symétrique est un graphe orienté symétrique.
La fermeture symétrique du graphe de la figure 2.1se trouve dans la figure 2.5.

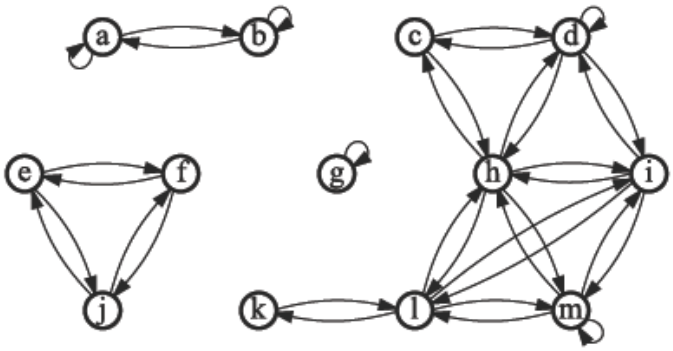


Figure 2.5 – Graphe orienté symétrique qui est la fermeture symétrique du graphe de la figure 2.1

2.3.2 Graphes non-orientés :

Un graphe non-orienté *G* est un couple *(S, A)* où *S* est un ensemble fini d'éléments, encore appelés sommets, et où *A* est un ensemble de paires non-ordonnées d'éléments, appelées alors arêtes, qui est un sous-ensemble de $S \otimes S = \{\{x, y\} \subseteq S \mid x \neq y\}$.

Si $\{x, y\}$ est une arête du graphe non-orienté *G = (S, A)*, alors on dit que :
– l'arête $\{x, y\}$ est incidente aux sommets *x* et *y*,
– l'arête $\{x, y\}$ relie les sommets *x* et *y*,
– les sommets *x* et *y* sont les extrémités de l'arête$\{x, y\}$, et
– que les sommets *x* et *y* sont adjacents ou voisins.

Soient $a_1 = (x_1, y_1)$ et $a_2 = (x_2, y_2)$ deux arêtes du graphe *G* = (*S*, *A*). On dit que les arêtes a_1 et a_2 sont adjacentes si $a_1 \cap a_2 \neq \emptyset$.

Par définition d'une arête, il ne peut y avoir d'arête reliant un sommet à lui-même. Pour tout sommet *x* d'un graphe non-orienté on définit le degré, noté *d(x)*, qui est le nombre d'arêtes incidentes au sommet *x* (là encore puisqu'il n'y a plus de notion d'ordre, il n'y a plus à faire de distinction comme dans un graphe orienté avec les degrés rentrant et sortant). On dit d'un sommet qu'il est isolé quand son degré est nul.

Dans un graphe non-orienté *G* = (*S*, *A*), on appelle chaîne du sommet x_0 au sommet x_ℓ dans *G* une séquence ordonnée de sommets $\pi = \langle x_0, \ldots, x_\ell \rangle$ de sommets de *S* tels que pour tout entier $k \in [1; \ell], \{x_{k-1}, x_k\} \in A$. On dit que la chaîne π contient les sommets $x_0, x_1, \ldots, x_\ell$ et les arêtes $\{x_0, x_1\}, \{x_1, x_2\}, \ldots, \{x_{\ell-1}, x_\ell\}$. Une chaîne peut également être définie par la séquence ordonnée des arêtes qui la constituent : $\pi = \langle a_0, \ldots, a_{\ell-1} \rangle$. On dit que ℓ est la longueur de la chaîne, c'est le nombre d'arêtes qu'elle contient.

Une chaîne est dite élémentaire si les sommets qui la constituent sont tous distincts. On dit que deux sommets *x* et *y* sont liés s'il existe une chaîne $\pi = \langle x, \ldots, y \rangle$ et, là encore, on peut facilement montrer que s'il existe une chaîne de *x* à *y*, alors il existe une chaîne élémentaire de *x* à *y*.

Une chaîne $\pi = \langle x_0, \ldots, x_\ell \rangle$est appelée cycle si $x_0 = x_\ell$, si elle contient au moins une arête et si elle ne contient pas deux fois la même arête. Si tous ses sommets sauf x_0 et x_ℓ sont distincts, on parle alors de cycle élémentaire. Un graphe qui ne contient aucun cycle est dit acyclique.

Exemple 2.2 :

Un exemple de graphe non-orienté se trouve dans la figure 2.6 (les sommets sont représentés par des cercles et les arêtes par des traits les reliant). Dans ce graphe :

- $d(h) = 5$;
- $\langle h, l, m, h, l, i, d, c \rangle$ est une chaîne de longueur *7* ;
- $\langle h, l, m, i, d, c \rangle$ est une chaîne élémentaire de longueur *5* ;
- $\langle e, f, j, e \rangle$ est un cycle élémentaire.

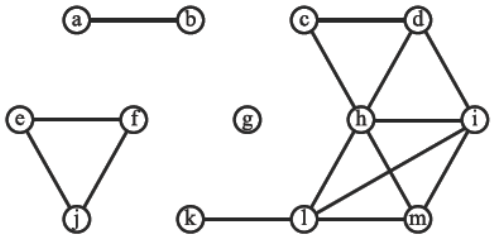


Figure 2.6 – Graphe non-orienté

2.3.3 Association entre graphes orientés et graphes non-orientés :

D'après les définitions énoncées précédemment, il est facile de voir que la donnée d'un graphe orienté symétrique antiréflexif est équivalente à celle d'un graphe non-orienté.

Cependant, il est parfois nécessaire, pour certaines définitions ou pour utiliser certains algorithmes, de devoir passer d'un graphe orienté (non-symétrique) à un graphe non-orienté. Pour ce faire, on considère le résultat de la fermeture symétrique du graphe orienté auquel on a préalablement supprimé les éventuelles boucles (afin de le rendre antiréflexif s'il ne l'était pas déjà). La donnée du graphe orienté obtenu est alors équivalente à celle d'un graphe non-orienté.
Voir l'exemple de graphe orienté de la figure 2.1 et sa fermeture symétrique, représentée sur la figure 2.5, qui sont tous deux associés au graphe non-orienté de la figure 2.6.

2.4 Notions communes et relations entre graphes orientés et non-orientés :

Qu'un graphe *G* soit orienté ou non, la notation *S*(*G*) fait référence à son ensemble de sommets et *A*(*G*) fait référence à son ensemble d'arcs ou d'arêtes. Ainsi on a *G* = (*S*(*G*), *A*(*G*)).

2.4.1 Sous-graphe :

Un sous-graphe d'un graphe $G = (S, A)$ est un graphe $G' = (S', A')$ où $S' \subseteq S$ et $A' \subseteq A$.

Soit $S' \subseteq S$. Le sous-graphe de G induit par S', noté $G_{S'}$, est le sous-graphe ayant pour ensemble de sommets S' et dont l'ensemble d'arcs (ou d'arêtes) A' est le sous-ensemble de A dont les extrémités appartiennent à S'. On dit que A' est la restriction de A à l'ensemble de sommets S' ou encore que A' est l'ensemble d'arcs (ou d'arêtes) de A qui sont induit(e)s par S'.

Soit $A' \subseteq A$. Le sous-graphe de G induit par A', noté $G_{A'}$, est le sous-graphe ayant pour ensemble d'arcs (ou d'arêtes) A' et dont l'ensemble de sommets S' est le sous ensemble de S qui sont des extrémités des éléments de A'. On dit que S' est la restriction de S à l'ensemble d'arcs (ou d'arêtes) de A' ou encore que S' est l'ensemble de sommets de S qui sont induits par A'.

La notation $G_\emptyset$ désigne le graphe vide, c'est-à-dire $G_\emptyset = (\emptyset, \emptyset)$, qui est un sous graphe de n'importe quel graphe.

2.4.2 Notations ensemblistes appliquées aux graphes :

Voici quelques notations et définitions concernant les graphes. Soient X et Y deux graphes.

– Le graphe X est inclus dans le graphe Y, noté $X \subseteq Y$, si $S(X) \subseteq S(Y)$ et $A(X) \subseteq A(Y)$, autrement dit si X est un sous-graphe de Y.

– Le graphe X est inclus strictement dans le graphe Y, noté $X \subset Y$, si $X \subseteq Y$ et $X \neq Y$, autrement dit X est un sous-graphe strict de Y.

– L'union des graphes X et Y, noté $X \cup Y$, est le graphe $(S(X) \cup S(Y), A(X) \cup A(Y))$.

– L'intersection des graphes X et Y, noté $X \cap Y$, est le graphe $(S(X) \cap S(Y), A(X) \cap A(Y))$.

– La différence (ensembliste) des graphes X et Y, noté $X \setminus Y$ (prononcé X "*moins*" Y), est le graphe induit par $A(X) \setminus A(Y)$ auquel on rajoute les sommets isolés de X qui ne sont pas dans Y.

Des exemples de ces définitions se trouvent dans la figure 2.7.

2.4.3 Connexité et composantes connexes :

Un graphe non-orienté G est connexe si pour toute paire de sommets $x, y \in S(G)$, x et y sont liés par une chaîne dans G.

Un graphe orienté G est faiblement connexe si pour toute paire de sommets $x, y \in S(G)$, x et y sont liés par une chaîne dans le graphe non-orienté G' associé à G.

Un graphe orienté G est fortement connexe si pour toute paire de sommets $x, y \in S(G)$, il existe un chemin de x à y dans G et il existe un chemin de y à x dans G.

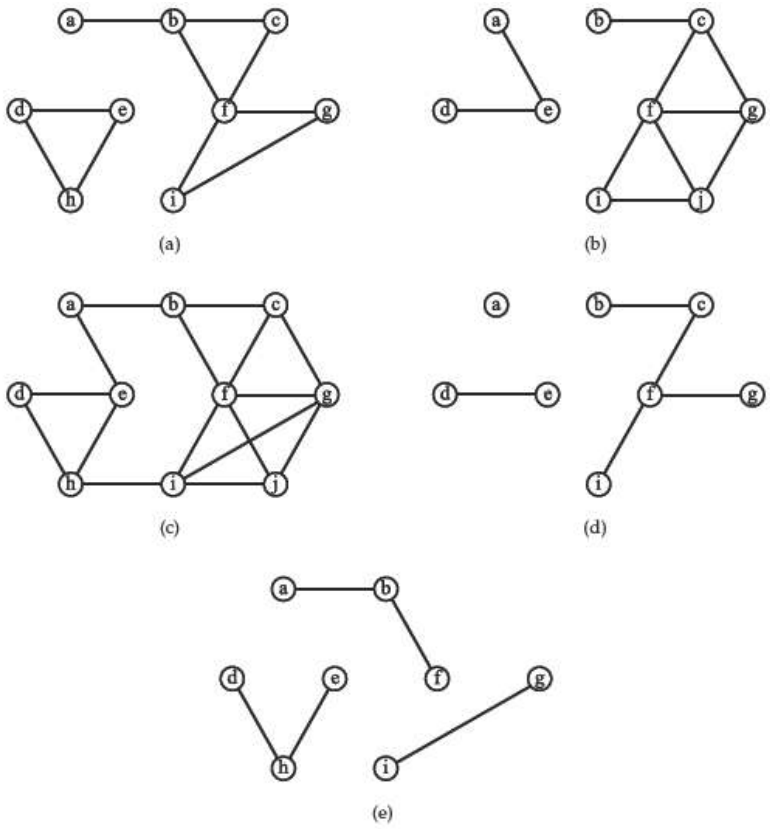


Figure 2.7 – (a) Graphe non-orienté G_1 ; (b) Graphe non-orienté G_2 ; (c) $G_1 \cup G_2$; (d) $G_1 \cap G_2$; (e) $G_1 \setminus G_2$.

Remarque 2.3 :

Le graphe $(x, \emptyset)$, qui est un sommet isolé, et le graphe vide $G_\emptyset$ sont fortement connexes.

Une composante connexe d'un graphe (orienté ou non-orienté) *G* est un sous-graphe connexe de *G* qui est maximal, autrement dit il n'existe pas de sous-graphe connexe de *G* qui l'inclut strictement.

L'ensemble des composantes connexes distinctes d'un graphe *G* se note $\mathcal{C}(G)$. Ainsi, $|\mathcal{C}(G)|$ représente le nombre de composantes connexes de *G*.

Si un graphe *G* est connexe, alors on a $\mathcal{C}(G) = \{G\}$, autrement dit, *G* ne possède qu'une seule composante connexe.

2.4.4 Complémentaire dans un graphe :

Dans un graphe $G = (S, A)$, le complémentaire d'un ensemble de sommets $S' \subseteq S$, noté $\overline{S'}$, est l'ensemble $S \setminus S'$, Respectivement, le complémentaire d'un ensemble d'arcs ou d'arêtes $A' \subseteq A$, noté $\overline{A'}$, est l'ensemble $A \setminus A'$. Ainsi, lorsque l'on parle de complémentaire d'un ensemble d'éléments dans un graphe, implicitement, l'ensemble référence est celui du graphe considéré.

Remarque 2.4 :

Soient un graphe $G = (S, A)$ et deux sous-ensembles $S' \subseteq S$ et $A' \subseteq A$. Notons $G_{S'}$ et $G_{A'}$ les sous-graphes induits respectivement par S' et A', $G_{\overline{S'}}$ et $G_{\overline{A'}}$ les sous-graphes induits respectivement par $\overline{S'}$ et $\overline{A'}$.

Il est à noter que $G_{A'} \cup G_{\overline{A'}} = G$ alors que $G_{S'} \cup G_{\overline{S'}}$ n'est, en général, pas égal à G, et également que $G_{S'} \cap G_{\overline{S'}} = G_{\emptyset}$ alors que $G_{A'} \cap G_{\overline{A'}}$ n'est, en général, pas égal à $G_{\emptyset}$.

2.4.5 Adjacence et fonction successeur :

Dans un graphe G, la relation d'adjacence sur les sommets peut être représentée par une application G de S dans $\mathcal{P}(S)$ qui associe à tout sommet s_1 de S l'ensemble :

$* \ \Gamma(s_1) = \{s_2 \in S \mid (s_1, s_2) \in A\}$ pour les graphes orientés (d'où l'équivalence $(s_1, s_2) \in A \Leftrightarrow s_2 \in \Gamma(s_1)$), et

$* \ \Gamma(s_1) = \{s_2 \in S \mid \{s_1, s_2\} \in A\}$ pour les graphes non-orientés (d'où la double équivalence $\{s_1, s_2\} \in A \Leftrightarrow s_2 \in \Gamma(s_1) \Leftrightarrow s_1 \in \Gamma(s_2)$ puisque les arêtes sont des ensembles).

Ainsi, un graphe G pourra également être représenté par le couple (S, Γ).

Dans le cas des graphes orientés, l'application Γ s'appelle fonction successeur puisque, à tout sommet du graphe G, elle fait correspondre l'ensemble de ses successeurs. De plus, le graphe orienté G^{-1}, symétrique du graphe orienté $G = (S, \Gamma)$, pourra alors être représenté par (S, Γ^{-1}). La fonction Γ^{-1} est appelée fonction prédécesseur puisque, à tout sommet du graphe G, elle fait correspondre l'ensemble de ses prédécesseurs.

La fonction successeur peut être étendue aux sous-graphes. Ainsi, pour un sous graphe X de G, on définit $\Gamma(X) = \cup \{\Gamma(x) \mid x \in X\} \setminus X$. La terminologie relative à l'adjacence peut ainsi être également utilisée en se référant à des sous-graphes. Ainsi, par exemple, un sommet y sera voisin d'un sous-graphe X s'il existe un arc/une arête a entre les sommets x et y avec $x \in S(X)$. Un arc

$a = (x, y)$/une arête $a = \{x, y\}$ sera adjacent(e) à *X* si $x \in S(X)$ et $a \notin A(X)$. Si $y \notin S(X)$, on dira alors que *a* est strictement adjacent(e).

De façon similaire à l'application *Γ*, la relation d'adjacence sur les arcs ou les arêtes peut être représentée par une application *γ* de *A* dans $\mathcal{P}(A)$ qui associe :

– à tout arc $a_1 = (x_1, y_1)$ de *A* l'ensemble $\gamma(a_1) = \{a_2 = (x_2, y_2) \in A \mid y_1 = x_2\}$ pour les graphes orientés (d'où l'équivalence $y_1 = x_2 \Leftrightarrow a_2 \in \gamma(a_1)$), et

– à toute arête $a_1 = \{x_1, y_1\}$ de A l'ensemble $\gamma(a_1) = \{a_2 = \{x_2, y_2\} \in A \mid a_1 \cap a_2 \neq \emptyset\}$ pour les graphes non-orientés (d'où la double équivalence $a_1 \cap a_2 \neq \emptyset \Leftrightarrow a_2 \in \gamma(a_1) \Leftrightarrow a_1 \in \gamma(a_2)$ puisque les arêtes sont des ensembles).

2.4.6 Clique et graphe complet :

Soit un graphe *G* = (*S*, *A*).

On appelle clique un sous-ensemble de sommets $S' \subseteq S$ tel que pour tout couple *(x, y)* de sommets de *S'*, *x* soit adjacent à *y* dans *G*. Ce qui peut également s'écrire :

– dans le cas des graphes orientés, $\forall x, y \in S', (x, y) \in A \; et \; (y, x) \in A$;

– dans le cas des graphes non-orientés, $\forall x, y \in S', \{x, y\} \in A$.

Une clique est maximale s'il n'existe pas de clique dans *G* qui la contienne strictement.

Exemple 2.5 :

Dans la figure 2.6, le singleton $\{g\}$ et l'ensemble $\{a, b\}$ sont des cliques maximales.

Un graphe est dit complet si l'ensemble de ses sommets forme une clique.

Des exemples de cette définition se trouvent dans la figure 2.8.

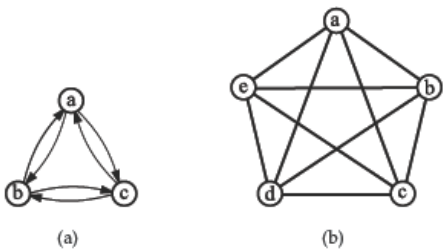


Figure 2.8 – (a) Graphe complet orienté à 3 sommets. (b) Graphe complet non-orienté à 5 sommets.

Il est à noter que la recherche de cliques maximales est un problème d'optimisation mis en évidence en 1972 par Richard Manning Karp comme étant NP-complet dans [KARP72] (où il présentait 21 problèmes NP-complets).

2.4.7 Graphes isomorphes :

Deux graphes, $G_1 = (S_1, A_1)$ et $G_2 = (S_2, A_2)$, sont dits isomorphes, noté $G_1 \sim G_2$, s'il existe une bijection f, appelée isomorphisme, entre S_1 et S_2 telle que deux sommets x et y de S_1 sont adjacents dans G_1 si et seulement si leurs images $f(x)$ et $f(y)$ de S_2 sont également adjacents dans $G2$.

Remarque 2.6 :

Un isomorphisme entre $G_1 = (S_1, A_1)$ et $G_2 = (S_2, A_2)$ induit une bijection entre A_1 et A_2.
La relation d'isomorphisme est une relation d'équivalence.

2.4.8 Graphe dérivé :

En 1932, Hassler Whitney fut le premier à introduire la notion de graphe dérivé (derived graph ou line graph en anglais). Celle-ci ne concernait alors que les graphes non-orientés comme le montre la définition 2.7 ci-dessous, mais elle fut ensuite étendue aux graphes orientés (voir définition 2.13). Même si cette dernière définition peut également servir pour le cas des graphes non-orientés, la définition 2.7 est plus aisée à comprendre pour l'introduction de ce concept.

2.4.8.1 Graphe dérivé non-orienté :

Définition 2.7 (Graphe dérivé non-orienté) **:**

Le graphe $G' = (S', A')$ est le graphe dérivé du graphe $G = (S, A)$ si :
– il existe une application bijective de A dans S', et
– chaque paire de sommets de S' est une arête de A' si et seulement si leurs arêtes correspondantes dans A sont adjacentes (autrement dit, si elles ont une extrémité commune).

On note $\partial(G)$ le graphe dérivé de G.

Ainsi, chaque arête d'un graphe dérivé représente une chaîne de longueur 2 dans le graphe d'origine.
Des exemples de graphes dérivés non-orientés se trouvent dans la figure 2.9.

Remarque 2.8 :

Le passage d'un graphe à son graphe dérivé permet ainsi de transférer les propriétés d'arêtes du graphe d'origine sur les sommets du graphe dérivé.

La construction du graphe dérivé G' de G implique que pour chaque sommet s de $S(G)$ il existe une clique dans $S(G')$ comprenant tous les sommets dérivants des arêtes de $A(G)$ qui sont incidentes à s. On en déduit qu'un graphe G' est le graphe dérivé d'un graphe G si et seulement si il est possible de trouver une collection de cliques maximales dont les arêtes induites forment une partition de $A(G')$ et telle que chaque sommet de G' appartienne à exactement deux de ces cliques. Bien entendu, certaines de ces cliques peuvent très bien n'être composées que d'un seul sommet. Hassler Whitney [WHITN32] prouva qu'il ne peut y avoir qu'une seule partition de ce type possible (au graphe "triangulaire" près - voir la remarque 2.10).
Ceci se retrouve à travers la propriété 2.9.

Propriété 2.9 :

Soit un graphe $G = (S, A)$.
Si pour tout sommet $s \in S$, s appartient à deux cliques maximales distinctes au plus, alors G est un graphe dérivé.

Remarque 2.10 :

Hassler Whitney montra qu'un graphe "triangulaire" (figures 2.9a et 2.9b) s'avère être une exception à la règle énoncée plus haut. En effet, ce graphe est en soi une clique maximale et devrait donc, si on le considère comme un graphe dérivé "classique", être représenté par un sommet isolé dans le graphe d'origine. Or, par construction, un sommet isolé disparaît lorsque l'on passe au graphe dérivé. Il faut donc alors considérer, non pas la clique maximale, mais trois cliques composées chacune d'une arête, permettant ainsi de retrouver le graphe d'origine associé.

Si un graphe est un graphe dérivé, il est alors possible de retrouver le graphe d'origine en créant un sommet pour chaque clique maximale et relier deux de ces sommets s'il existe un sommet dans le graphe dérivé appartenant aux deux cliques correspondantes (voir figure 2.10), à l'exception toujours du graphe "triangulaire".

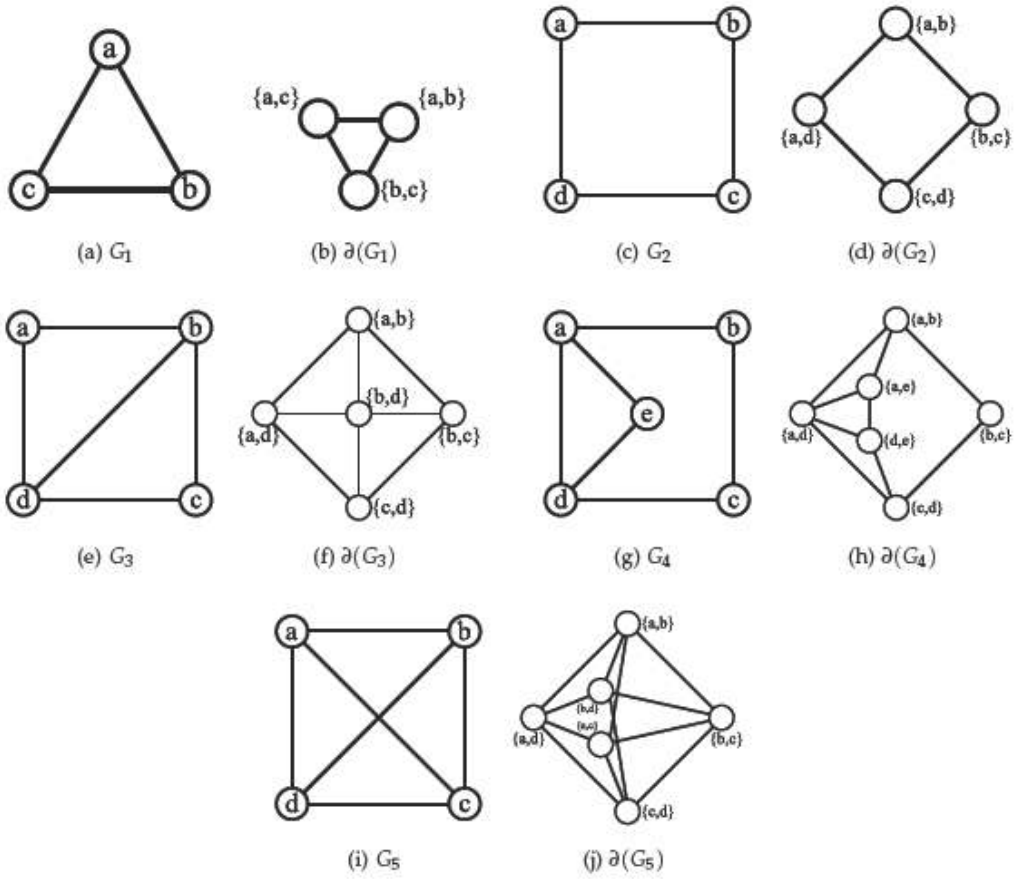


Figure 2.9 – Des graphes non-orientés (a, c, e, g, i) et leurs graphes dérivés non-orientés respectifs (b, d, f, h, j).

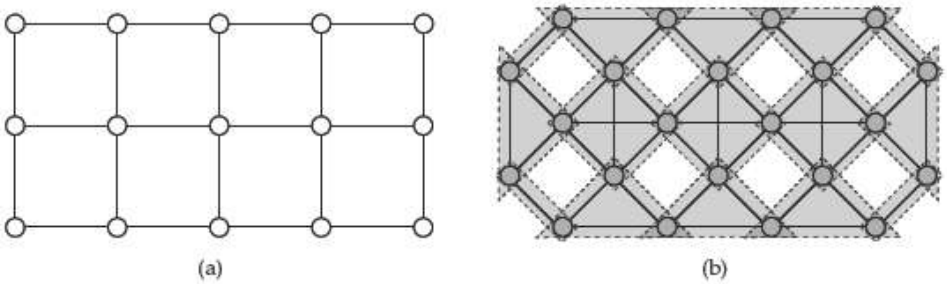


Figure 2.10 – (a) Graphe non-orienté *G* ; (b) Graphe non-orienté *G'* dérivé de *G*. Les cliques maximales de *G'* sont représentées dans les zones grisées. Chacun des sommets de *G'* appartient à exactement deux cliques maximales.

En se basant sur l'unicité de cette construction, Nicholas D. Roussopoulos, en 1973 [ROUSSO73], et Philippe G. H. Lehot, en 1974 [LEHOT74], proposèrent des algorithmes optimaux linéaires permettant de vérifier si un graphe est un graphe dérivé et, le cas échéant, de construire son graphe d'origine.

En 1995, une version dynamique de ces algorithmes fut établie dans [DEGIOR95], permettant ainsi, lorsque l'on ajoute ou supprime un sommet dans un graphe dérivé, de mettre à jour son graphe d'origine en temps proportionnel aux nombres de sommets qui lui sont adjacents.

Une caractérisation alternative des graphes dérivés fut prouvée par Lowell W. Beineke en 1968 [BEINEK68]. Il démontra qu'il existe neuf graphes minimaux qui ne sont pas des graphes dérivés, de telle sorte que tout graphe qui n'est pas un graphe dérivé contient un de ces graphes en tant que sous-graphe induit (voir théorème 2.11).

Théorème 2.11 : ([BEINEK68])

Un graphe *G* est un graphe dérivé si et seulement si aucun de ses sous-graphes induits par un sous-ensemble de sommets n'est isomorphe à l'un des neuf graphes de la figure 2.11.

Remarque 2.12 :

Le fait de prendre en considération les sous-graphes induits par un sous-ensemble de sommets et non pas simplement les sous-graphes est important. En effet, il est possible qu'un graphe dérivé ait pour sous-graphe (non induit par un sous-ensemble de sommets) un des neuf graphes cités dans le théorème 2.11 (voir figure 2.12).

2.4.8.2 Graphe dérivé orienté :

Définition 2.13 (Graphe dérivé orienté) **:**

Le graphe $G' = (S', A')$ est le graphe dérivé du graphe $G = (S, A)$ si :

– il existe une application bijective de A dans S', et

– tout couple de sommets (s'_1, s'_2) de S', représentant respectivement les arcs $a_1 = (x_1, y_1)$ et $a_2 = (x_2, y_2)$ de A, est un arc de A' si et seulement si a_2 est adjacent à a_1 (i.e. si $y_1 = x_2$).

On note $\partial(G)$ le graphe dérivé de G.

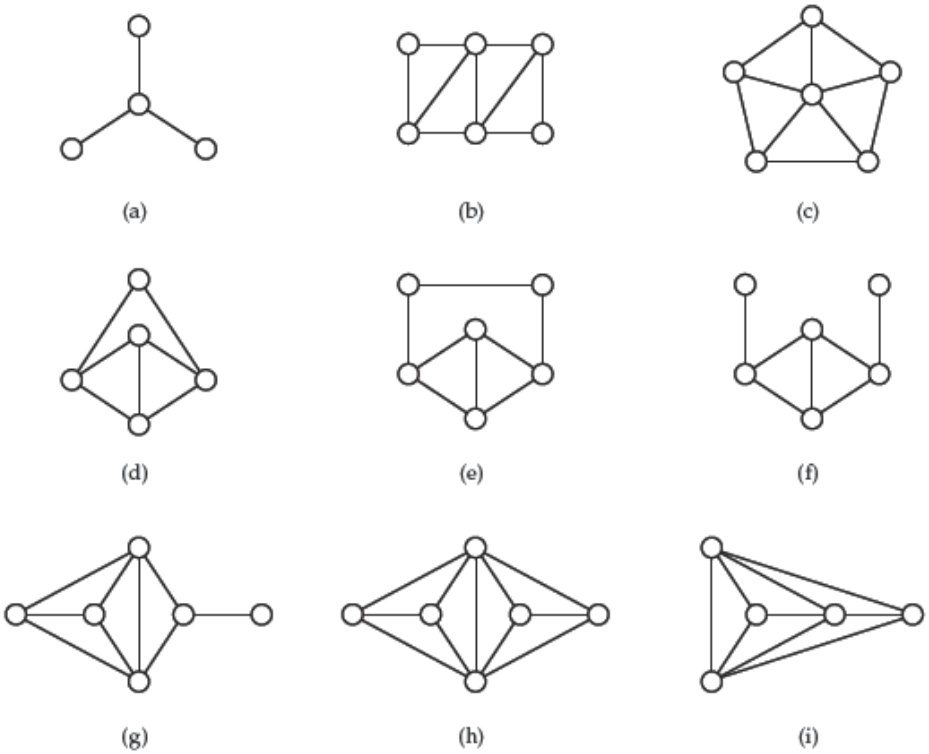


Figure 2.11 – Les neuf graphes minimaux non-dérivés.

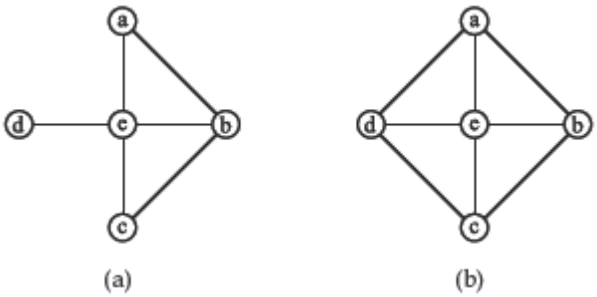


Figure 2.12 – (a) Graphe non-dérivé car ayant pour sous-graphe induit par les sommets {*a, c, d, e*} un des graphes non-dérivés minimaux (Figure 2.11a) ; (b) Graphe dérivé (voir figures 2.9e et 2.9f) malgré qu'il contienne en sous-graphe un des graphes non-dérivés minimaux (figure 2.11a).

Ainsi, de manière similaire au cas non-orienté, chaque arc d'un graphe dérivé représente un chemin de longueur 2 dans le graphe d'origine.

Des exemples de graphes dérivés orientés se trouvent dans la figure 2.13.

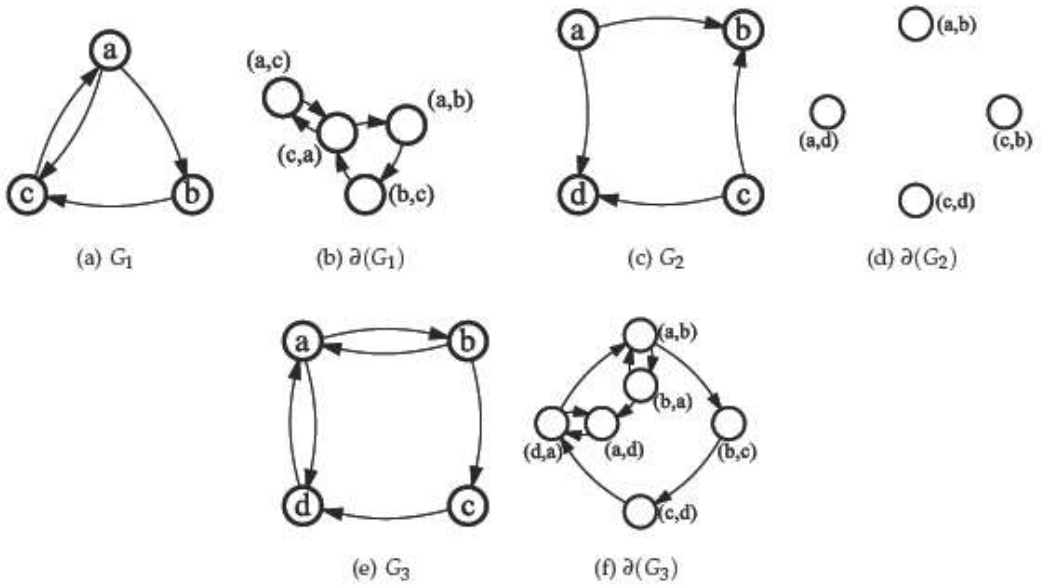


Figure 2.13 – Des graphes orientés (a, c, e) et leurs graphes dérivés orientés respectifs (b, d, f).

2.4.8.3 Notations et remarques sur les graphes dérivés orientés ou non :

On dit que le graphe *G* est un graphe dérivé s'il existe un graphe *G'* tel que *G* est isomorphe au graphe dérivé de *G'*.

Dans la suite de cette sous-section, nous considérerons le graphe $G' = (S', A')$ comme le dérivé du graphe $G = (S, A)$.

Remarque 2.14 :

Par construction, les arcs/arêtes a_1 et a_2 de *A* sont adjacent(e)s dans *G* si et seulement si leurs sommets dérivés respectifs s_1 et s_2 de *S'* sont adjacents dans *G'*.

Si *a* est un arc ou une arête de *G*, alors le sommet dérivé de *a* (pour *G*), noté $\partial_G(a)$ ou simplement $\partial(a)$ lorsqu'il n'y a pas de doute sur le graphe concerné par la dérivation, est le sommet de *G'* auquel l'application bijective considérée lors de la dérivation fait correspondre *a*.

De manière similaire, si *s* est un sommet de *G*, alors l'ensemble d'arcs/arêtes dérivé(e)s de *s* (pour *G*), noté $\partial_G(s)$ ou simplement $\partial(s)$ lorsqu'il n'y a pas de doute sur le graphe concerné par la dérivation, est l'ensemble d'arcs/arêtes *A''* de *G'* qui représentent les adjacences se faisant par le biais de *s* dans *G*. Ces arcs/arêtes sont induit(e)s par une clique maximale de $\partial(G)$.

Remarque 2.15 :

Soit un sommet *s* de *G*.

Il est à noter que si $d^-(s) = 0$ ou $d^+(s) = 0$, pour le cas des graphes orientés, ou bien $d(s) \leq 1$, pour le cas des graphes non-orientés, alors $\partial(s) = \emptyset$.

Propriété 2.16 :

Soit *X* un sous-graphe de *G* qui n'a qu'une composante connexe.

Si *X* n'est pas un sommet isolé, alors il existe un sous-graphe $X' = (\partial(A(X)), \partial(S(X)))$ dans *G'* qui est connexe.

Preuve de la propriété 2.16. La remarque 2.14 nous permet de déduire que l'ensemble de sommets $\partial(A(X))$ est bien connexe dans *G'*. De plus, par définition, les extrémités des arcs/arêtes de $\partial(S(X))$ sont toutes incluses dans $\partial(A(X))$.

Le sous-graphe $(\partial(A(X)), \partial(S(X)))$ de *G'* est donc bien connexe.

Grâce à la proposition 2.16, la notation de dérivation peut ainsi s'étendre à des sous graphes. Si *X* est un sous-graphe de *G*, alors le sous-graphe dérivé de *X* (pour *G*), noté $\partial_G(X)$ ou simplement $\partial(X)$ lorsqu'il n'y a pas de doute sur le graphe concerné par la dérivation, est le sous-graphe *X'* de *G'* tel que $X' = \left(\partial\left(A(X)\right), \partial\left(S(X)\right)\right)$. Bien sûr, si $X = (s, \emptyset)$ alors $\partial(X) = G_\emptyset$.

Propriété 2.17 :

Si *X* est un sous-graphe de *G* ne possédant pas de sommet isolé, alors le nombre de composantes connexes du sous-graphe dérivé $\partial(X)$ est égal au nombre de composantes connexes de *X* (i.e. $|C(\partial(X))| = |C(X)|$).

2.4.9 Poids sur un graphe :

Dans beaucoup d'algorithmes ou d'applications, que le graphe soit orienté ou non, il s'avère nécessaire d'associer des valeurs aux éléments de ce graphe. La pondération d'un graphe se fait grâce à une application de poids *P*, qui peut être de deux types :

– $S \to \mathcal{E}$, on a alors un graphe pondéré sur les sommets, aussi appelé graphe à sommets values, et $\forall s \in S, P(s)$ est le poids, ou l'altitude, du sommet *s* ;

– $A \to \mathcal{E}$, on a alors un graphe pondéré sur les arcs/arêtes, aussi appelé graphe à arcs/arêtes valué(e)s, et $\forall a \in A, P(a)$ est le poids, ou l'altitude, de l'arc/arête a.

L'ensemble de pondération $\mathcal{E}$ peut être quelconque et est souvent choisi en fonction de l'application dans laquelle le graphe pondéré intervient. Voici quelques exemples typiques : $\mathbb{Z}, \mathbb{R}, [0; 255]$ ou encore $[0; 255]^3$.

Le poids d'un graphe *G* est :

– la somme des poids des sommets de *G* si *G* est un graphe pondéré sur les sommets, noté

$$\mathrm{P(G)} = \sum_{\mathrm{s \in S(G)}} \mathrm{P(s)}; \qquad (2.1)$$

– la somme des poids des arcs/arêtes de G si G est un graphe pondéré sur les arcs/arêtes, noté

$$\mathrm{P(G)} = \sum_{\mathrm{a \in A(G)}} \mathrm{P(a)}; \qquad (2.2)$$

Remarque 2.18 :

La notion de graphe dérivé présentée plus haut permet de passer, dans le cas de graphes non orientés, d'un graphe à arêtes valuées à un graphe dérivé à sommets valués en transférant simplement l'application de poids *P* de l'ensemble d'arêtes du graphe original à l'ensemble de sommets du graphe dérivé. En conséquence de la remarque 2.8, le poids du graphe reste alors inchangé. Voir les Figures 2.14a et 2.14b.

Il est à noter que l'on peut également considérer le passage, toujours dans le cas de graphes non orientés, d'un graphe à sommets valués à un graphe dérivé à arêtes valuées en transférant le poids de chaque sommet du graphe initial à chaque arête de la clique qui le représente dans le graphe dérivé. Mais dans ce cas, on se rend compte que le poids du graphe est supérieur. Voir les Figures 2.14c et 2.14d.

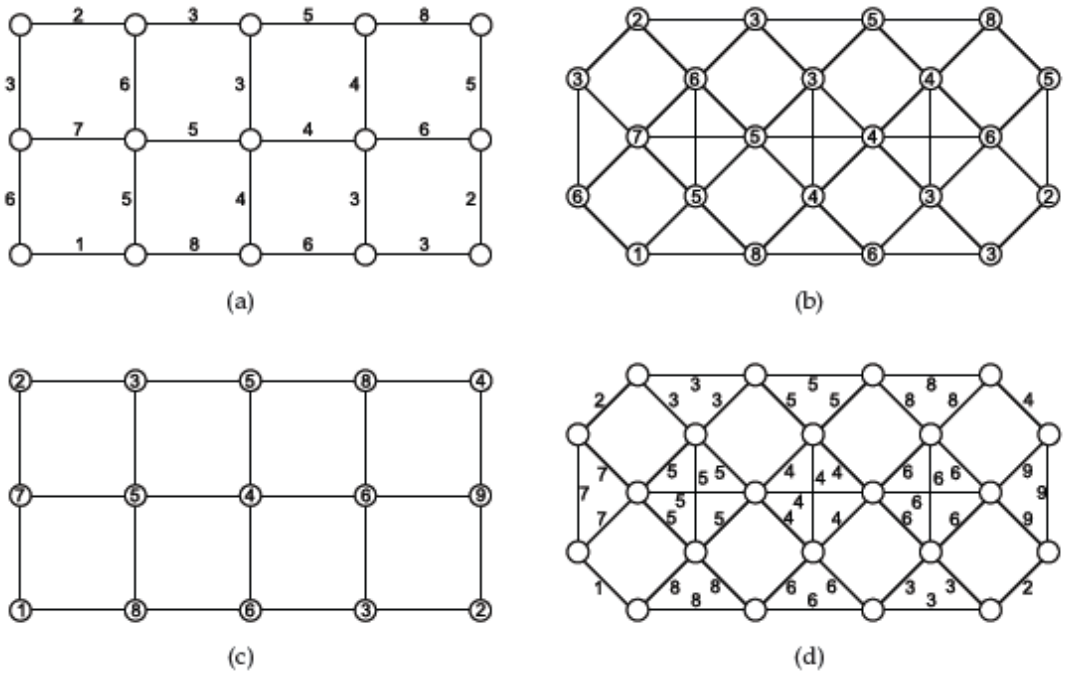


Figure 2.14 – (a) Graphe G_1 à arêtes valuées ; (b) Graphe G'_1 dérivé de G_1 à sommets valués ; (c) Graphe G_2 à sommets valués ; (d) Graphe G'_2 dérivé de G_2 à arêtes valuées.

Dans la suite de cette section, afin de simplifier la lecture des prochaines définitions et d'unifier dans une même notation les cas des graphes à sommets valués et celui des graphes à arcs/arêtes valué(e)s, nous noterons E l'ensemble sur lequel se fait la pondération d'une application de poids P sur un graphe $G = (S, A)$. Ainsi :

– si G est un graphe à sommets valués, $E = S$;

– si G est un graphe à arêtes valuées, $E = A$.

Cette notation s'applique également aux sous-graphes. Ainsi, en considérant X un sous graphe de G, on a :

– si G est un graphe à sommets valués, $E(X) = S(X)$;

– si G est un graphe à arêtes valuées, $E(X) = A(X)$.

Cela nous permettra ainsi de n'avoir qu'une seule définition lorsque les deux cas peuvent être considérés de façon semblable.

2.5 Graphes usuels en traitement d'images :

Il est courant de voir la notion de graphe, ainsi que les algorithmes qui l'accompagnent, utilisé dans le cadre du traitement d'images.

Dans cette section, nous présenterons tout d'abord comment peut se définir une image avant d'expliquer comment les différents éléments qui la composent peuvent être associés aux éléments d'un graphe.

2.5.1 Qu'est-ce qu'une image numérique ?

Une image numérique I est une application dont le domaine de définition, noté $\mathcal{D}(I)$ est discret. Nous nous limiterons ici au cas où $\mathcal{D}(I) \subseteq \mathbb{Z}^n$ qui est le plus usuel. Typiquement, celui-ci est de la forme $[0; d_1] \times [0; d_2] \times \ldots \times [0; d_n]$ et n représente la dimension de l'image.

Dans le cas où $n = 2$, qui est le plus courant, on a ainsi une image bidimensionnelle (2D) dans laquelle chaque élément de $\mathbb{Z}^2$ est représenté par un carré, aussi appelé pixel, centré sur les coordonnées correspondantes, dont les côtés sont parallèles aux axes et ont une longueur de 1.

De manière analogue, dans le cas où $n = 3$, on a une image tridimensionnelle (3D) dans laquelle chaque élément de $\mathbb{Z}^3$ est représenté par un cube, aussi appelé voxel, centré sur les coordonnées correspondantes, dont les arêtes sont parallèles aux axes et ont une longueur de 1.

Il est à noter que le cas où $n = 4$ fait souvent référence à une séquence temporelle d'images 3D (on parle alors de "3D + temps").

Le domaine d'application de l'image est, le plus souvent, lui aussi discret mais également borné avec, typiquement :

– [0;255] pour une image en niveaux de gris (avec la valeur 0 pour le noir et 255 pour le blanc) ;

– $[0;255]^3$ pour une image en couleurs RVB (Rouge, Vert, Bleu) où chacun des canaux est échelonné de la même façon qu'une image en niveaux de gris décrite ci-dessus.

Il est à noter que la borne supérieure de ces exemples courants a été choisie par convention en fonction de la taille d'un octet et donc du nombre de valeurs qu'il peut stocker (en effet, un octet étant composé de 8 bits, il y a donc $2^8 = 256$ valeurs possibles).

2.5.2 Voisinage dans une image :

Lorsque l'on fait un traitement sur une image, il faut souvent lui adjoindre une définition du voisinage à considérer. En effet, dans le cas 2D par exemple, il est nécessaire de préciser si l'on considère qu'un pixel a pour voisins les pixels qui ont un côté en commun avec lui uniquement ou bien également ceux qui partagent avec lui un sommet. Bien que les deux cas soient valables, un même algorithme pourra donner des résultats très différents selon que l'on soit dans un cas ou dans

l'autre. En conséquence de quoi il a été défini plusieurs voisinages types déterminant si deux n-uplets $x = (x_1, \ldots, x_n)$ et $y = (y_1, \ldots, y_n)$ sont voisins dans $\mathbb{Z}^n$. Ainsi, on a :

– en 2D ($n = 2$) :

– dans le 4-voisinage, x et y sont voisins si

$$\sqrt{(x_1 - y_1) + (x_2 - y_2)} \leq \sqrt{1} = 1 \qquad (2.3)$$

(deux pixels sont voisins s'ils ont un côté en commun, ainsi chaque pixel non au bord de l'image a 4 pixels voisins) ;

– dans le 8-voisinage, x et y sont voisins si

$$\sqrt{(x_1 - y_1) + (x_2 - y_2)} \leq \sqrt{2} \qquad (2.4)$$

(deux pixels sont voisins s'ils ont au moins un coin en commun, ainsi chaque pixel non au bord de l'image a 8 pixels voisins, ce qui est le maximum).

– en 3D ($n = 3$) :

– dans le 6-voisinage, x et y sont voisins si

$$\sqrt{(x_1 - y_1) + (x_2 - y_2) + (x_3 - y_3)} \leq \sqrt{1} = 1 \qquad (2.5)$$

(deux voxels sont voisins s'ils ont une face en commun, ainsi chaque voxel non au bord de l'image a 6 voxels voisins) ;

– dans le 18-voisinage, x et y sont voisins si

$$\sqrt{(x_1 - y_1) + (x_2 - y_2) + (x_3 - y_3)} \leq \sqrt{2} \qquad (2.6)$$

(deux voxels sont voisins s'ils ont au moins une arête en commun, ainsi chaque voxel non au bord de l'image a 18 voxels voisins) ;

– dans le 26-voisinage, x et y sont voisins si

$$\sqrt{(x_1 - y_1) + (x_2 - y_2) + (x_3 - y_3)} \leq \sqrt{3} \qquad (2.7)$$

(deux voxels sont voisins s'ils ont au moins un sommet en commun, ainsi chaque voxel non au bord de l'image a 26 voxels voisins, ce qui est le maximum).

– en nD (cas général avec $n > 0$), soit un entier $d \in [0; n]$ et soit l'entier V_d^n tel que :

– si $d = 1$, alors $V_d^n = 2n$;

– si $d = n$, alors $V_d^n = 3^n - 1$;

– si $1 < d < n$, alors $V_d^n = V_d^{n-1} + 2 \times (V_{d-1}^{n-1} + 1)$;

dans le V_d^n voisinage, x et y sont voisins si

$$\sqrt{(x_1 - y_1) + (x_2 - y_2) + \cdots + (x_n - y_n)} \leq \sqrt{d} \qquad (2.8)$$

(chaque élément non au bord de l'image a V_d^n éléments voisins).

Exemple 2.19 :

Si l'on considère l'image *2*D de taille 3×3 de la figure 2.15 associée au *4*-voisinage, alors le pixel *e à* pour voisins les pixels $\{b, d, f, h\}$. Par contre, si elle est associée au *8*-voisinage, alors le pixel *e à* pour voisins les pixels $\{a, b, c, d, f, g, h, i\}$.

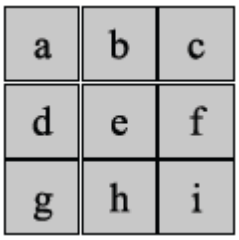


Figure 2.15 – Image 2D de taille 3×3 (3 pixels par 3 pixels).

2.5.3 Image, voisinage et graphe :

La définition de voisinage entre éléments d'une image (entre pixels typiquement) n'est pas sans rappeler celle d'adjacence entre les sommets d'un graphe. Nous mettons en évidence ci-après que l'introduction de voisinages dans les images permet de se retrouver dans le contexte des graphes. En effet, une image *I* associée à un voisinage est un graphe non-orienté *G* pondéré sur les sommets. On a alors les implications suivantes :

– Chaque élément de $\mathcal{D}(I)$ (autrement dit, chaque pixel d'une image 2D, par exemple) est un sommet de *G*.
– Tout couple $\{x, y\}$ de sommets tels que *x* et *y* sont voisins est une arête dans *G*.
– L'application qui pondère les sommets est *I*.

On appelle grille (de dimension n) tout graphe *(S, A)* (orienté ou non) tel que $S \subseteq \mathbb{Z}^n$.
Typiquement, on a $S = [0; d_1] \times [0;\ d_2] \times \ldots \times [0;\ d_n]$.
Par définition, le graphe auquel correspondent une image et son voisinage est une grille.

On appelle V_d^n adjacence la relation d'adjacence dans une grille équivalente au V_d^n voisinage de l'image *I*. Nous noterons ainsi $A_d^n(S)$, ou plus simplement A_d^n , l'ensemble d'arêtes correspondant à la V_d^n adjacence pour *S*. Autrement dit $\{x, y\} \in A_d^n$ si et seulement si $x, y \in S$ et que *x* et *y* sont V_d^n adjacents.

Il est à noter qu'il est tout à fait possible, lorsque cela est nécessaire pour un certain algorithme par exemple, de placer les poids du graphe correspondant à l'image sur les arêtes au lieu des sommets. Il suffit alors de prendre une fonction adéquate au type d'utilisation que l'on souhaite faire du graphe (par exemple, le poids de l'arête peut être la valeur maximale des deux pixels correspondants aux sommets qui sont à ses extrémités).

Des exemples de représentations de graphes issus d'une image 2D en niveaux de gris associée à différents voisinages se trouvent dans la figure 2.16.

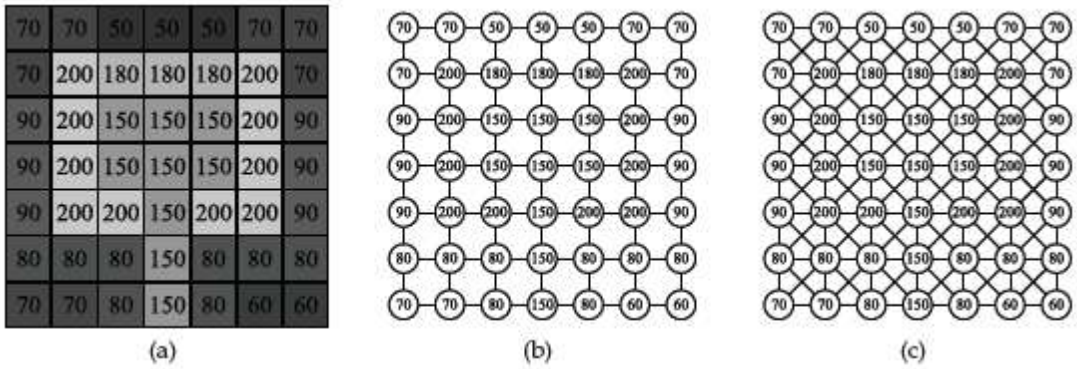


Figure 2.16 – (a) Une image en niveaux de gris. (b) Le graphe correspondant à l'image de la figure (a) associé au 4-voisinage. (c) Le graphe correspondant à l'image de la figure (a) associé au 8-voisinage.

2.6 Conclusion :

Outre le fait que ce chapitre nous a permis de poser les bases qui seront nécessaires à la bonne compréhension des notions abordées dans les chapitres suivants, nous avons mis en évidence l'équivalence entre images et certains graphes, les grilles. Sans cette équivalence, de nombreux traitements d'images ne pourraient exister. Prenons l'exemple de la segmentation d'une image en deux parties : un objet et son arrière-plan. Ne serait-ce que dans la description de cette tâche, nous faisons implicitement appel à des définitions relatives aux graphes. En effet, ceci implique les notions d'adjacence, de sous-graphe et de composantes connexes.

CHAPITRE 3 :

Coupe minimale et flot maximal

3.1 Introduction :

Nous présentons dans ce chapitre les concepts de réseaux de transport et de flot maximal liés à la recherche de coupe minimale qui constitue une technique de segmentation de graphe.

Un réseau de transport est un graphe orienté pondéré sur les arcs et dans lequel on distingue deux sommets particuliers : la source et le puits. De manière informelle, cela se conçoit en imaginant un matériau qui transite via un réseau de transport (tel qu'un réseau routier ou des canalisations) d'une source où il est produit vers un puits où il est consommé. Ainsi, chaque arc du graphe peut être vu comme un moyen de transport ou un conduit que doit emprunter le matériau pour se rendre de son sommet de départ vers son sommet d'arrivée. La quantité de matériau qui peut ainsi transiter d'un sommet à un autre par unité de temps est limitée par la taille du conduit ou par le moyen de transport qu'il emprunte (par exemple, cinq ampères de courant électrique dans un câble ou bien vingt litres d'eau par seconde dans une canalisation). Ceci est représenté par la capacité de l'arc, qui est, dans la cadre des réseaux de transport, un autre terme pour désigner le poids d'un arc. Les sommets du graphe représentent donc les jonctions, les carrefours, entre ces conduits. Excepté pour la source et le puits, le matériau transite via ces jonctions sans gain ni perte (ce qui correspond à la loi de Kirchhoff dans un réseau électrique). La recherche d'un flot maximal dans un réseau de transport vise à déterminer la quantité maximale de matériau qui peut transiter à travers lui.

Les notions traitées dans ce chapitre concernent principalement les graphes orientés. C'est pourquoi nous considérons le graphe orienté connexe $G = (S, A)$. La pondération du graphe est systématiquement une application de poids de valeurs positives sur les arcs ou les arêtes telle que $P : A \rightarrow \mathbb{R}^+$.

Il est à noter que, dans la suite de ce mémoire, il nous arrivera de considérer un poids infini, noté $+\infty$, sur un arc ou une arête. En pratique, cela représente simplement un poids très supérieur à la somme des poids finis du graphe considéré.

3.2 Réseau de transport et flot :

3.2.1 Réseau de transport :

Un réseau de transport est un graphe orienté à arcs valués par l'application de poids $P: A \rightarrow \mathbb{R}^+ \cup \infty$ dans lequel on choisit deux sommets distincts ayant un statut particulier :

– la source, notée *s*, et

– le puits, noté *p*.

De plus, tout sommet du graphe appartient à un chemin reliant la source au puits : $\forall x \in S, \exists \pi = \langle s, \ldots, x, \ldots, p \rangle$. En conséquence de quoi *G* est connexe. Il est à noter que le poids de chaque arc pourra également être appelé, dans ce cadre, capacité étant donné qu'il représente la quantité maximale de matériau pouvant transiter par l'arc qu'il pondère.

Par la suite, nous noterons $\mathcal{R}$ = *(S, A, P, s, p)* le réseau de transport sur *G* pondéré par *P* associé à la source *s* et au puits *p*. Un exemple de réseau de transport se trouve dans la figure 3.1.

3.2.2 Flot :

On appelle flot (compatible avec le réseau de transport $\mathcal{R}$ = *(S, A, P, s, p)*) une application, notée *f*, de A^* dans $\mathbb{R}$ ($f : A^* \rightarrow \mathbb{R}$; nous rappelons que $A^* = A \cup A^{-1}$, ce qui correspond à l'ensemble d'arcs de la fermeture symétrique du graphe *G = (S, A)'* voir section 2.3.1.2), qui satisfait les trois propriétés suivantes :

– **Contrainte de capacité** :

si $a \in A$, alors $f(a) \leq P(a)$ (3.1)

et si $a \in A^* \setminus A$, alors $f(a) \leq 0$ (3.2)

– **Symétrie** :

si $a \in A^*$, alors $f(a) = -f(a^{-1})$ (3.3)

– **Conservation du flot** :

si $x \in S \setminus \{s, p\}$, alors

$$\sum_{a=(x,.),a\in A^*} f(a) = 0. \quad (3.4)$$

La quantité *f(a)* est appelé flot net de l'arc *a*. Bien entendu, la propriété de contrainte de capacité indique que le flot net d'un arc ne peut excéder sa capacité.

La propriété de symétrie est une convention mettant en relation directe un arc et son symétrique de sorte à connaître le flot qui transite entre deux sommets en regardant l'un ou l'autre des deux arcs les reliant dans A^*.

Remarque 3.1 :

Il est à noter que si l'arc *a* et son symétrique a^{-1} sont tous deux dans *A* et qu'on fait transiter en "réalité" un flot f_a dans *a* et un flot f_{a-1} dans a^{-1}, nous aurons pour flots nets $f(a) = f_a - f_{a-1}$ et $f(a^{-1}) = f_{a-1} - f_a$.

C'est dans ce cas que la notion de flot net négatif prend tout son sens.

Enfin, la propriété de conservation du flot indique bien évidemment que le flot entrant en un sommet est égal au flot sortant de ce sommet puisque le transit s'y fait sans gain ni perte. On observe ici l'utilité du flot net négatif dans cette notation.

On définit alors :

– le flot positif entrant en un sommet $x \in S$:

$$f^+(x) = \sum_{a=(.,x),a\in A,f\ (a)>0} f(a) \qquad (3.5)$$

– le flot positif sortant en un sommet $x \in S$:

$$f^-(x) = \sum_{a=(x,.),a\in A,f\ (a)>0} f(a) \qquad (3.6)$$

Et bien sûr, pour tout sommet $x \in S \setminus \{s, p\}$, on déduit de la propriété de conservation du flot :

$$f^+(x) = f^-(x) \qquad (3.7)$$

Remarque 3.2 :

Il est à noter que la représentation d'un flot, de par sa définition, doit se faire sur la fermeture symétrique (voir la définition en section 2.3.1.2) du graphe du réseau de transport. Cependant, pour des raisons de clarté de représentation, on peut se contenter de ne montrer que le flot net positif, qui est donc représenté sur les arcs du graphe d'origine uniquement.

Voir des exemples sur la figure 3.1.

On appelle valeur du flot *f*, notée $|f|$, la quantité totale de flot net partant de la source et arrivant au puits :

$$|f| = \sum_{a=(s,.),a\in A^*} f(a) = \sum_{a=(.,p),a\in A^*} f(a) \qquad (3.8)$$

Des exemples de flots compatibles avec un réseau de transport se trouvent dans la figure 3.1.

Remarque 3.3 :

Il est à noter qu'un réseau $\mathcal{R} = (S, A, P, s, p)$ est équivalent à un réseau $\mathcal{R}' = (S, A \setminus \{a\}, P, s, p)$ où a est un arc de poids nul dans $\mathcal{R}$.

Ci-après se trouve la définition d'un flot particulier de grand intérêt.

Définition 3.4 (Flot maximal) **:**

Un flot compatible avec le réseau de transport $\mathcal{R}$ est dit maximal, s'il n'existe aucun flot compatible avec le réseau $\mathcal{R}$ de valeur strictement supérieure.

La recherche d'un flot maximal s'avère utile dans de multiples applications qui nécessitent de faire transiter des matériaux dans un réseau, comme par exemple :

– de l'eau dans des canalisations,

– du courant dans un réseau électrique,

– des données sur un réseau informatique, ...

La section suivante met toutefois en évidence que la valeur d'un flot maximal n'est pas la seule donnée intéressante à en retirer.

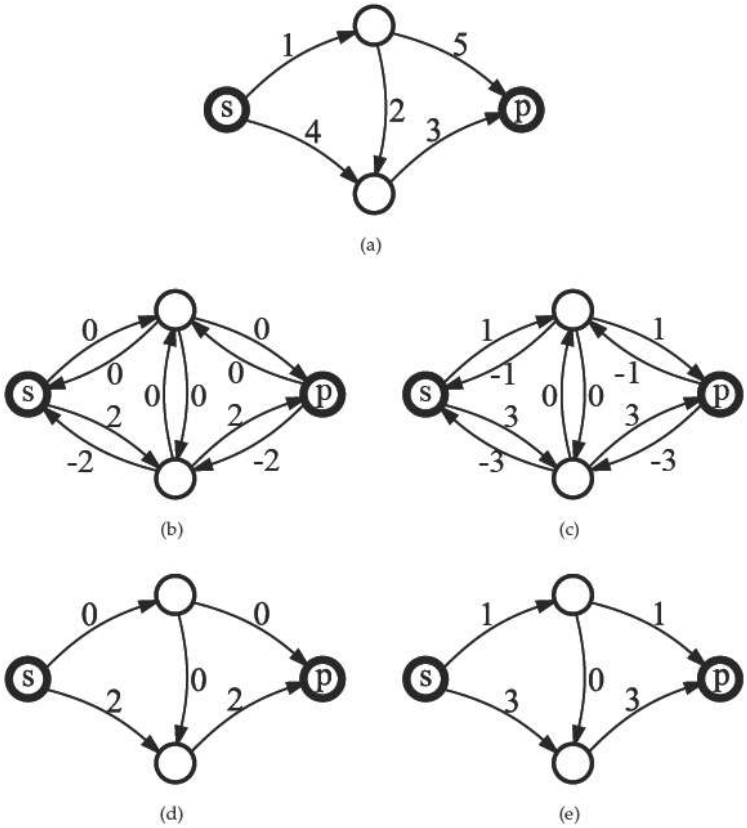


Figure 3.1 – (a) Réseau de transport $\mathcal{R}$ avec la source s et le puits p représentés en gras. (b) Flot f_1 compatible avec $\mathcal{R}$. (c) Flot maximal f_2 compatible avec $\mathcal{R}$. (d) Flot net positif de f_1. (e) Flot net positif de f_2.

3.3 Lien entre flot maximal et coupe minimale :

Comme il vient d'être montré, le calcul d'un flot maximal a de nombreuses applications, mais, dans le cadre de la segmentation de graphe, ce sont les coupes minimales qui nous intéresseront. Ce qui suit présente le lien qui unit flot maximal et coupe minimale.

Définition 3.5 (Coupe d'un réseau de transport) :

Une coupe (d'un réseau de transport) est un ensemble d'arcs $C \subseteq A$ qui peut être défini de deux façons équivalentes :

– il n'existe aucun chemin reliant la source *s* au puits *p* dans le graphe $G' = (S, A \setminus C)$ et aucun sous-ensemble strict $C' \subset C$ ne vérifie cette propriété ;

– $C = \{a = (x, y) \mid x \in \mathcal{S} \; et \; y \in \mathcal{P}\}$ avec $\{\mathcal{S}, \mathcal{P}\}$ une partition de S telle que $s \in \mathcal{S}$, $p \in \mathcal{P}$ et les graphes induits respectivement par $\mathcal{S}$ et par $\mathcal{P}$ soient connexes.

Il est à noter que l'ensemble d'arêtes correspondant à une coupe d'un réseau de transport dans le graphe non-orienté associé à ce réseau de transport n'est pas nécessairement une coupe, comme le montre la figure 3.2. Dans le cas où le graphe du réseau de transport est symétrique, nous obtenons néanmoins une équivalence entre ces deux types de coupes, comme le montre la propriété 3.6.

Propriété 3.6 :

Soit $\mathcal{R} = (S, A, P, s, p)$ un réseau de transport dont le graphe $G = (S, A)$ est symétrique. Soit $G' = (S, A')$ le graphe non-orienté associé à *G*.

L'ensemble d'arcs *C* est une coupe du réseau de transport $\mathcal{R}$ si et seulement si l'ensemble d'arêtes *C'* qui lui est associé est une coupe relative à $\{\{s, p\}, \emptyset\}$ sur *G'*.

Une illustration de cette propriété se trouve dans la figure 3.2.

Remarque 3.7 :

Outre la propriété 3.6 qui lie coupe relative à un sous-graphe et coupe d'un réseau (définition 3.5), il est à noter que cette dernière définition permet également de procéder à la segmentation d'un graphe en offrant une partition des sommets du graphe en deux régions.

Par définition, le poids d'une coupe *C* d'un réseau de transport $\mathcal{R}$ vaut :

$$P(C) = \sum_{a \in C} P(a) \qquad (3.9)$$

On appelle coupe minimale d'un réseau de transport une coupe de poids minimal.
Le théorème suivant, prouvé dans [DANTZIG56] et présenté dans [DIESTEL97], montre le lien qui unit la recherche d'une coupe minimale avec la recherche d'un flot maximal.

Théorème 3.8 :
Dans un réseau de transport, la valeur d'un flot maximal est égale au poids d'une coupe minimale.

Ce théorème implique que si l'on sait trouver un flot maximal d'un réseau, on peut également en déduire une coupe minimale. La détermination des arcs appartenant à cette coupe sera mise en évidence plus loin dans le théorème 3.10.

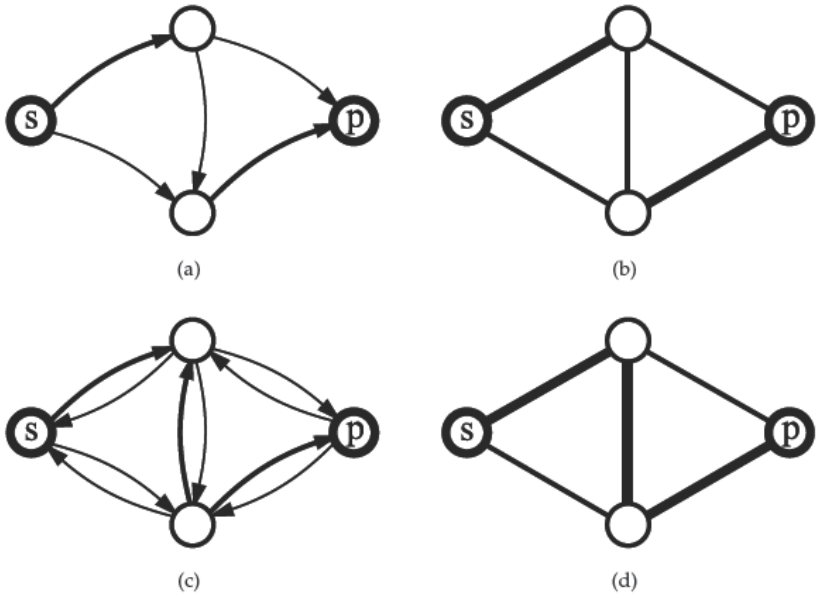


Figure 3.2 – (a) Réseau $\mathcal{R}_1$ = *(S, A₁, P₁, s, p)* avec, en gras, une coupe C_1. (b) Graphe non-orienté associé à $\mathcal{R}_1$ avec, en gras, les arêtes associées à C_1 qui ne sont pas une coupe relative à $(\{s, p\}, \emptyset)$. (c) Réseau $\mathcal{R}_2$ = *(S, A₂, P₂, s, p)* avec, en gras, une coupe C_2. (d) Graphe non-orienté associé à $\mathcal{R}_2$ avec, en gras, les arêtes associées à C_2 qui sont bien une coupe relative à $(\{s, p\}, \emptyset)$.

Remarque 3.9 :

Il est à noter que si l'on modifie la capacité des arcs d'un réseau $\mathcal{R}$ = *(S, A, P, s, p)* pour obtenir un réseau $\mathcal{R}'$ = *(S, A, P', s, p)* tel que, pour toute coupe *C* dans $\mathcal{R}$ (et donc également dans $\mathcal{R}'$), *P'(C)=P(C)+K* où *K* est une constante, alors une coupe C^* est minimale pour $\mathcal{R}$ si et seulement si elle est minimale pour $\mathcal{R}'$.

De plus, si l'on note f^* un flot maximal pour $\mathcal{R}$ et f'^*un flot maximal pour $\mathcal{R}'$, alors nous avons $|f'^*| = |f| + K$.

La section suivante présente les algorithmes les plus connus permettant d'obtenir un flot maximal dans un réseau.

3.4 Recherche de flot maximal :

Comme nous l'avons vu dans la section 3.3, la recherche d'une coupe minimale peut se faire à travers la recherche d'un flot maximal. Un tour d'horizon des algorithmes de calcul de flots maximaux existant en 1998 se trouve dans l'article [GOLDBERG91]. Les livres [COOK98, SCHRIJVER03] peuvent également être recommandés pour une étude plus détaillée de ces méthodes.

3.4.1 Recherche de flot maximal par chemins améliorants :

Le premier algorithme de recherche de flot maximal, développé en 1956 par Lester Randolph Sr Ford et Delbert Ray Fulkerson [FORD62], est à l'origine de cette première grande catégorie d'algorithmes. Son principe repose sur des améliorations successives du flot, ce dernier étant initialisé par n'importe quel flot compatible (un flot de valeur nulle étant compatible avec tout réseau de transport, c'est souvent celui-ci qui sert de flot de départ). Pour ce faire, on cherche itérativement des chemins reliant la source au puits qui vont permettre d'accroître ce flot.

Tout d'abord, nous présenterons les définitions nécessaires à la description de l'algorithme de Ford et Fulkerson pour finir par une brève présentation des variantes améliorant les temps de calcul de cet algorithme.

3.4.1.1 Réseau résiduel :

Soient un réseau de transport $\mathcal{R}$ = *(S, A, P, s, p)* et un flot *f* compatible avec $\mathcal{R}$.

On appelle capacité résiduelle d'un arc $a = (x, y) \in A^*$ (pour le flot *f* dans un réseau de transport $\mathcal{R}$), notée *r(a)*, la quantité de flot supplémentaire qu'il est possible de faire transiter "en plus" entre *x* et *y*. Plus précisément :

$$r(a) = \begin{cases} P(a) - f(a) & si\ a \in A; \\ -f(a) & sinon. \end{cases} \qquad (3.10)$$

On dit qu'un arc $a \in A$ est saturé (par le flot *f* dans le réseau de transport $\mathcal{R}$) si *f (a) = P(a)*, autrement dit, si on ne peut plus *y* faire croître le flot sans perdre la compatibilité avec le réseau. Sa capacité résiduelle est donc nulle *(r(a) = 0)*.

On appelle réseau résiduel (du flot *f* dans le réseau de transport $\mathcal{R}$) le réseau $\mathcal{R}_f$ = *(S, A_f, r, s, p)* où :

$$A_f = \{a \in A^* \mid r(a) > 0\} \qquad (3.11)$$

Ainsi, un réseau résiduel représente l'ensemble des arcs pouvant permettre d'augmenter le flot.

Un arc *a* ne peut apparaître dans le réseau résiduel que si $a \in A$ ou $a^{-1} \in A$. Il est ainsi possible qu'un arc *a* soit dans A_f mais pas dans *A*. Un tel arc apparaît dans $\mathcal{R}_f$ seulement si $a^{-1} \in A$ et si $f(a^{-1}) > 0$. Il représente la possibilité de réduire le flot qui transite par a^{-1} afin qu'il soit "redirigé" vers d'autres arcs partant du sommet d'arrivée de *a* (voir un exemple sur la figure 3.5e).

Des exemples de réseaux résiduels se trouvent dans la figure 3.3.

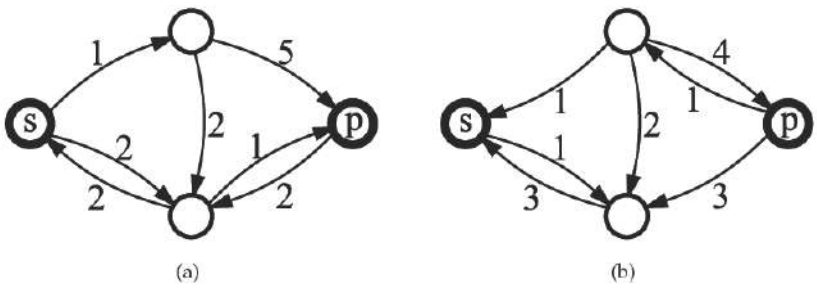


Figure 3.3 – (a) Graphe résiduel de f_1 sur $\mathcal{R}$ (figure 3.1). (b) Graphe résiduel de f_2 sur $\mathcal{R}$ (figure3.1). La source et le puits sont représentés en gras.

La définition de réseau résiduel donnée plus haut nous permet de déterminer les arcs appartenant à une coupe minimale pour un flot maximal donné comme le montre le théorème 3.10.

Théorème 3.10 :

Un arc $a \in A$ appartient à une coupe minimale C d'un réseau de transport $\mathcal{R} = (S, A, P, s, p)$ si et seulement si, dans le graphe résiduel d'un flot maximal, son sommet de départ est dans la composante connexe contenant la source et son sommet d'arrivée dans la composante connexe contenant le puits.

La définition d'arc saturé donnée plus haut nous permet de définir un nouveau type de flot particulier à comparer au flot maximal.

Définition 3.11 (Flot complet) :

Un flot compatible avec le réseau de transport $\mathcal{R}$ est dit complet, ou bloquant, si tout chemin de la source au puits passe par au moins un arc saturé.

Le théorème qui suit montre le lien entre flot complet et flot maximal.

Théorème 3.12 :

Un flot maximal est un flot complet.

Il est à noter que la réciproque du théorème 3.12 n'est, en général, pas vraie. Une illustration de cela se trouve dans la figure 3.4.

3.4.1.2 Chemin améliorant :

On appelle chemin améliorant, ou chemin augmentant, du réseau $\mathcal{R}$ pour le flot f un chemin élémentaire reliant la source s au puits p dans le réseau résiduel $\mathcal{R}_f$ de f dans $\mathcal{R}$.

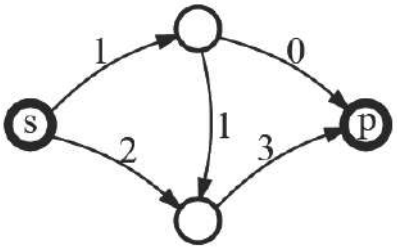


Figure 3.4 – Flot complet, mais non maximal, compatible avec le réseau de transport de la figure 3.1a.

Cette définition met en évidence la nécessité de construire le graphe résiduel d'un flot. En effet, nous avons vu que la réciproque du théorème 3.12 est fausse, en conséquence de quoi, même si tout chemin reliant la source au puits dans le réseau passe par au moins un arc saturé, un chemin améliorant peut tout de même exister dans le graphe résiduel.

On appelle capacité résiduelle d'un chemin améliorant π de $\mathcal{R}$ pour *f*, notée *r(π)*, la quantité de flot qu'il est possible d'ajouter à f sur les arcs de π en gardant la propriété de conservation du flot et la compatibilité avec $\mathcal{R}$. Plus précisément :

$$r(\pi) = \min_{a \in \pi}\{r(a)\} \qquad (3.12)$$

3.4.1.3 Algorithme de Ford et Fulkerson :

L'algorithme de Ford et Fulkerson [FORD62] consiste à partir d'un flot *f* compatible avec le réseau de transport $\mathcal{R}$ et à itérer les étapes suivantes :

1. Chercher un chemin améliorant π dans $\mathcal{R}$ pour *f*. Si un tel chemin n'existe pas, l'algorithme se termine et le flot courant est un flot maximal. Sinon, passer à l'étape 2.
2. Mettre à jour le flot *f* en augmentant les arcs de π de la valeur *r(π)*. Retourner à l'étape 1.

Un exemple de déroulement de cet algorithme se trouve dans la figure 3.5.

Le temps d'exécution de cet algorithme dépend fortement de l'heuristique choisie pour trouver le chemin améliorant de l'étape 1. Si celle-ci est mal choisie et que les capacités du réseau de transport ne sont pas des nombres rationnels, l'algorithme peut ne jamais se terminer, le flot augmentant progressivement sans pour autant converger vers un flot maximal. Un exemple de ce type se trouve dans [SCHRIJVER03].

Le plus souvent, cet algorithme est utilisé avec des capacités entières. On obtient alors une complexité en $O(|A|\left|\tilde{f}\right|)$ au pire où $\tilde{f}$ est un flot maximal (à chaque itération, on cherche un chemin, en $O(|A|)$, qui permet d'augmenter le flot d'au moins 1 unité). Il est à noter que lorsque les capacités du réseau sont des nombres rationnels, une réduction au même dénominateur permet de travailler malgré tout avec des valeurs entières.

3.4.1.4 Variantes et améliorations :

Comme dit précédemment, la stratégie de sélection du chemin améliorant est déterminante pour la rapidité d'exécution de l'algorithme. C'est pourquoi des variantes de ce premier algorithme ayant des heuristiques de détermination de chemins améliorants virent le jour.

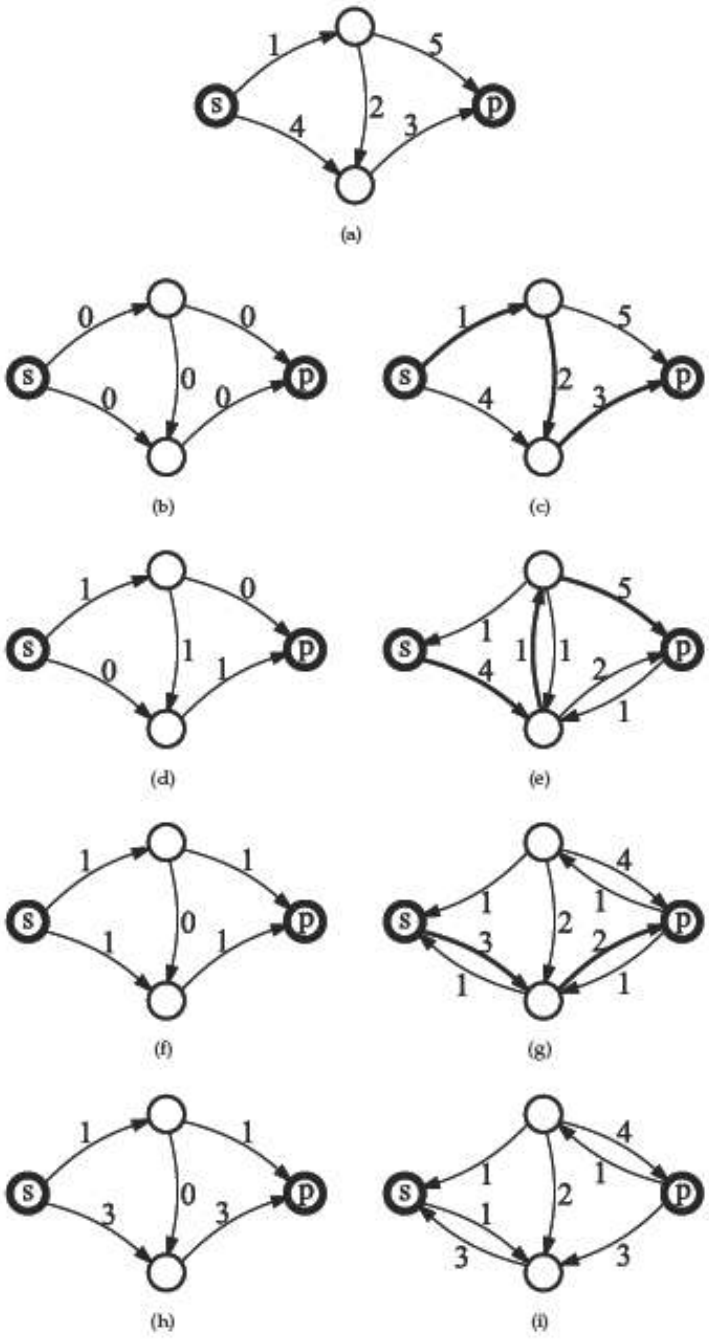


Figure 3.5 – (a) Réseau de transport $\mathcal{R}$. (b,d,f,h) Flots nets positifs de, respectivement f_0, f_1, f_2, f_3, compatibles avec $\mathcal{R}$. (c,e,g,i) Graphes résiduels de, respectivement f_0, f_1, f_2, f_3, sur $\mathcal{R}$. Dans (c,e,g), un chemin améliorant est représenté en gras.

En 1970, E. A. Dinic proposa un algorithme [DINIC70] basé sur une recherche de chemins améliorants par un parcours en largeur [CORMEN01]. Cette recherche permet ainsi de saturer tous les plus courts chemins d'une même longueur k avant de refaire un parcours en largeur du graphe résiduel et ainsi saturer les plus courts chemins d'une même longueur $k + 1$ à leur tour et ainsi de suite. La complexité de cet algorithme est en $O(|S|^2|A|)$.

En 1972, Jack Edmonds et Richard M. Karp présentèrent deux stratégies de sélection de chemin améliorant accélérant l'algorithme [EDMONDS72]. La première, ressemblant beaucoup à celle de E.A. Dinic, consiste à chercher les chemins améliorants les plus courts. La seconde consiste à chercher d'abord le chemin améliorant ayant la capacité résiduelle la plus grande. La complexité de ces algorithmes est en $O(|S||A|^2)$, moins bonne que celle de l'algorithme de Dinic puisque les chemins améliorants ne sont pas traités par groupes de même longueur.

En 2001, alors que depuis des années les algorithmes par poussée de flots étaient préférés, Yuri Boykov et Vladimir Kolmogorov proposèrent un nouvel algorithme ayant pour but d'être optimisé pour les utilisations dans lesquelles intervient ce type d'algorithme en vision par ordinateur [BOYKOV04]. Celui-ci se base sur l'algorithme de Dinic, reposant également sur la recherche de plus courts chemins, mais, pour ce faire, procède en créant non pas un, mais deux arbres de parcours en largeur : un partant de la source dans le graphe résiduel et un autre partant du puits dans le symétrique du graphe résiduel. Cependant, à la différence des algorithmes précédents, l'arbre de recherche n'est pas reconstruit à chaque itération, il est conservé et modifié en fonction du dernier chemin qui a été saturé. Cela a pour conséquence que le chemin choisi n'est pas toujours le plus court, la complexité devenant en $O(|S||A|^2|\tilde{f}|)$, le rendant théoriquement moins intéressant. Malgré cela, des tests sur les temps de calculs de diverses tâches en vision par ordinateur et en traitement d'images montrèrent qu'il était plus rapide car adapté au type de problème. C'est, depuis lors, cet algorithme qui fait office de référence en vision par ordinateur.

3.5 Coupe minimale dans un graphe non-orienté :

Comme expliqué dans la section (2.3.3), un graphe non-orienté pondéré sur les arêtes permet directement la recherche de coupes minimales puisqu'il est considéré équivalent à un graphe orienté symétrique ayant des poids identiques sur un arc et son arc symétrique.

Le théorème 3.8 peut ainsi facilement être étendu au cas des graphes non-orientés, comme montré dans [FORD62]. En effet, en se référant à la propriété 3.6, on se rend compte qu'il suffit de

considérer le graphe orienté symétrique associé avec, pour la pondération, la capacité de chaque arête utilisée sur les deux arcs symétriques qui lui sont associés et d'y appliquer n'importe quel algorithme de recherche de flot maximal afin de déterminer une coupe minimale.

Une illustration de cette équivalence entre coupe minimale sur graphe non-orienté et son graphe orienté symétrique associé se trouve dans la figure 3.6.

Remarque 3.13 :

Il est à noter que dans le cas d'une recherche de coupe minimale dans un graphe non-orienté, la source et le puits peuvent être inversés sans changement du résultat puisque le réseau de transport correspondant est un graphe symétrique avec les mêmes capacités sur un arc et son symétrique.

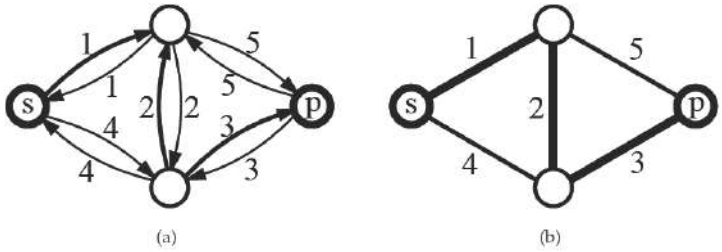


Figure 3.6 – (a) Réseau $\mathcal{R}$ = *(S, A, P, s, p)* avec, en gras, une coupe minimale *C*. (b) Graphe non-orienté associé à $\mathcal{R}$ avec, en gras, les arêtes associées à *C* qui sont bien une coupe minimale relative à $(\{s, p\}, \emptyset)$.

3.7 Conclusion :

Nous avons présenté dans ce chapitre ce qu'est un réseau de transport et un flot dans le cadre des graphes. Nous avons abordé ensuite la recherche de coupe minimale dans un réseau de transport en rappelant le résultat fondamental d'optimisation combinatoire qu'est l'équivalence entre celle-ci et la recherche d'un flot maximal. Après cela, nous avons présenté la façon de calculer un flot maximal, avec l'algorithme des chemins améliorants, celui de Ford et Fulkerson, qui furent les premiers à résoudre ce problème.

CHAPITRE 4 :

Segmentation d'images par coupure de graphe

4.1 Introduction :

En traitement d'images, de nombreux problèmes peuvent se mettre sous la forme de la minimisation d'une fonction d'énergie : la segmentation, la restauration, etc. En segmentation d'images, les méthodes variationnelles bien connues sont par exemple les contours actifs [KASS88], [COHEN91] et les approches par courbes de niveaux [SETHIAN99], [OSHER03].

Dans ce domaine, ces fonctions d'énergie intègrent en général un terme d'attache aux données incorporant des informations sur les contours de l'image (par exemple basées sur le gradient de l'image) et un terme de régularisation permettant de maîtriser l'aspect du contour. Cependant, l'optimisation peut poser problème lorsque la fonction d'énergie est non-convexe et présente des minima locaux. D'autre part, notons que dans ces approches, l'image est considérée comme un domaine continu et la nécessaire discrétisation de l'espace et du temps peut poser des problèmes de stabilité.

Offrant des solutions aux problèmes cités précédemment, les méthodes fondées sur les coupes de graphe ont suscité un fort intérêt depuis l'article fondateur de Boykov et Jolly [BOYKOV01]. En s'appuyant sur des algorithmes issus de la communauté combinatoire, elles permettent d'optimiser des fonctions de coût de manière globale, et ainsi d'éviter les minima locaux. D'autre part, en considérant l'image comme un graphe, la stabilité numérique est assurée. La segmentation d'images par coupe de graphe tel que présentée dans [BOYKOV01] autorise également de manière très souple une interaction avec l'utilisateur et permet d'obtenir une convergence rapide en 2D. Elle a donné lieu à de nombreux développements théoriques [KOLMO05] et applicatifs [ASLAN09] [BOYKOV00] [GARCIA09] [PUNDLIK10].

Dans les applications où l'objet à segmenter est connue a priori (par exemple un organe en imagerie médicale), le processus de segmentation peut être guidé par des contraintes sur celle-ci. Pour des images ayant un contraste faible ou un taux de bruit élevé, une telle contrainte permet d'améliorer la précision de la segmentation.

Dans ce chapitre, nous proposons une méthode de segmentation interactive basée sur les coupes de graphe binaires.

L'image segmentée est alors divisée en 2 parties:

- "l'objet d'intérêt" ;
- et le "background" (le fond de l'image ou l'arrière plan).

Au début, l'utilisateur impose des contraintes dures qui indiquent si un pixel appartient à l'objet ou fond. De plus, d'autres contraintes souples (fonction de cout ou énergie) incluent des informations sur les bords et les régions de l'image. L'objectif est alors de trouver une coupe optimale de graphe donnant une segmentation.

Notre travail est basé sur les travaux de [BOYKOV01] qui se distinguent par leur formalisme interactif, qui permettent l'extraction facile des pixels d'intérêt (objet ou arrière plan), en raison de l'implantation des germes par l'utilisateur. Dans le cadre du traitement d'images, le modèle (s-*p*) coupe utilise l'algorithme du flot maximum (Max Flow) de Ford et Fulkerson [FORD62] qui recherche une chaîne augmentante de la source (*s*) vers le puits (*p*) tels que les arcs de la chaîne ne soient pas saturés. Il est montré dans cette procédure que, pour tout couple (s, *p*) de sommets du graphe et pour toute pondération positive, la valeur maximum du flot de (*s*) à (*p*) est égale aussi à la capacité minimum d'une coupe séparant (*s*) à (*p*).

L'utilisation de ce résultat permet d'envisager une partition binaire optimale du graphe en minimisant une fonction d'énergie. Par ailleurs, en associant de la connaissance a priori au graphe, nous pouvons espérer obtenir une extraction d'objets d'intérêt suivant un processus de regroupement perceptuel.

4.2. Segmentation d'objets par la coupe de graphe :

4.2.1 Problème d'étiquetage ou labellisation des pixels :

De nombreux problèmes de traitement d'images ou de vision par ordinateur peuvent être vus sous la forme de problèmes d'étiquetage. La segmentation de graphes, par exemple, revient à assigner, pour chaque sommet du graphe considéré, une unique étiquette d'appartenance à telle ou telle région. Dans le cas de la segmentation d'images, ces étiquettes peuvent être du type : objet, arrière-plan,... De façon générale, le type des étiquettes dépend complètement de l'application dans laquelle intervient l'étiquetage.

Un problème d'étiquetage se définit par un ensemble de sites $\mathcal{S}$ et un ensemble d'étiquettes $\mathcal{L}$. Tous deux dépendent grandement du type d'application considérée. Un site représente souvent un point ou une région, tel que, par exemple, un pixel d'une image (qui est le cas le plus courant), une face d'un maillage par exemple. Une étiquette représente une caractérisation possible d'un site, tel que, par exemple, appartenir à une région donnée d'une image.

La résolution d'un problème d'étiquetage consiste à trouver, selon certains critères, une combinaison x qui assigne à chaque site i de $\mathcal{S}$ une unique étiquette $\ell = x_i$ de $\mathcal{L}_i$ qui est le sous-ensemble d'étiquettes de $\mathcal{L}$ qui peuvent être prises pour le site i. On forme ainsi une partition des sites $S = \{\mathcal{S}_\ell / \ell \in \mathcal{L}\}$ où $\mathcal{S}_\ell = \{i \in \mathcal{S} | x_i = \ell\}$ est une région de la partition.

Il est à noter que $\forall i \in \mathcal{S}$, $\mathcal{L}_i \neq \emptyset$ puisque tout site doit pouvoir être étiqueté. Cependant, si $|\mathcal{L}_i| = 1$ alors l'étiquette du site i est contrainte à une unique étiquette possible.
Nous pouvons néanmoins avoir une étiquette $\ell \in \mathcal{L}$ telle que $\mathcal{S}_\ell = \emptyset$, autrement dit cette étiquette n'est représentée sur aucun site.

Soit X l'ensemble des combinaisons possibles d'un problème d'étiquetage avec m sites et n étiquettes. Dans le pire des cas (i.e. quand tous les sites peuvent prendre n'importe quelle étiquette de $\mathcal{L}$), le nombre de ces combinaisons possibles est de $|X| = n^m$. Comme toute étiquette n'est pas nécessairement possible pour tout site, cela réduit le nombre de combinaisons possibles.

Cependant, leur nombre n'en demeure pas moins trop élevé pour permettre de tester ces combinaisons de manière exhaustive et trouver celle qui répond le mieux aux critères d'attribution des étiquettes. Des méthodes d'optimisation combinatoires furent ainsi développées afin de permettre une résolution de tels problèmes de manière plus efficace.

Dans beaucoup de cas applicatifs, le nombre d'étiquettes est égal à 2, soit $\mathcal{L} = \{0, 1\}$ où 1 correspond à l'objet d'intérêt et 0 à l'arrière plan de l'image. La stratégie pour classifier les pixels implique une recherche de solution optimale sur les suites de labels. Celle-ci se ramène à un problème de minimisation d'une fonction d'énergie qui est décrite au paragraphe ci-après.

4.2.2 Minimisation de la fonction d'énergie [BOYKOV01]:

L'objectif visé consiste à trouver une coupe optimale de graphes donnant une segmentation satisfaisante. La solution obtenue donne le meilleur équilibre entre les propriétés de bords et des régions parmi toutes les solutions respectant les contraintes.

Ce résultat implique à minimiser une fonction objective de la forme :

$$E(A) = \lambda\, R(A) + B(A) \tag{4.1}$$

Avec :

$$R(A) = \sum_{i \in S} R_i(A_i)$$

$$B(A) = \sum_{(i,j) \in \mathcal{V}} B_{(i,j)} \cdot \delta(A_i, A_j)$$

$$\delta(A_i, A_j) = \begin{cases} 1 & si\ A_i \neq A_j \\ 0 & sinon \end{cases}$$

où $\mathcal{V} \subset S \times S$ est un voisinage sur les pixels, le terme $R_i(A_i)$ est une fonction mesurant le coût d'attribution d'un label A_i à un pixel i, le terme $B_{ij}(A_i, A_j) \geq 0$ mesure le coût d'assignation des labels A_i et A_j aux pixels adjacents i et j, et le paramètre de pondération $\lambda \geq 0$ qui contrôle l'importance relative entre les termes de vraisemblance et d'a priori qui composent l'énergie ou en d'autres termes l'importance relative du terme "région" $R(A)$ par rapport au terme "contour ou bord" $B(A)$.

Dans le cas d'une coupe de graphes, la minimisation de l'énergie $E(A)$ pourra être obtenue de façon optimale si la condition de régularité [BOYKOV04] présentée ci-après est respectée :

$$E_{i,j}(0,0) + E_{i,j}(1,1) \leq E_{i,j}(0,1) + E_{i,j}(1,0) \tag{4.2}$$

4.2.3 Formulation du graphe dans l'image :

Un grand nombre d'auteurs considèrent que la segmentation d'images peut être vue comme un problème de labellisation [LI04, SHI00]. Nous décrivons dans ce paragraphe la construction d'un graphe à partir d'une image illustrée comme le montre la figure 4.1.

Chaque pixel de l'image est vu comme un nœud dans un graphe. Les arêtes *A* sont constituées de nœuds *S* adjacents et de leur capacité *P* dont la valeur dépend de la similarité entre deux pixels voisins.

Cette description peut être formalisée selon un graphe pondéré que l'on note *G = (S, A, P)*. Une grande (faible) capacité correspond à une faible (grande) similarité entre les pixels voisins.

Dans le cadre du modèle établi par Boykov et Jolly [BOYKOV01], deux nœuds terminaux particuliers représentant respectivement la classe objet (la source : *s*) et la classe arrière plan (le puits : *p*) terminent le graphe. Suivant les labels assignés par l'utilisateur lors du marquage des germes, les coûts des capacités sont calculés à l'aide des modèles définis pour l'objet ou l'arrière plan de l'image comme le décrit le paragraphe suivant.

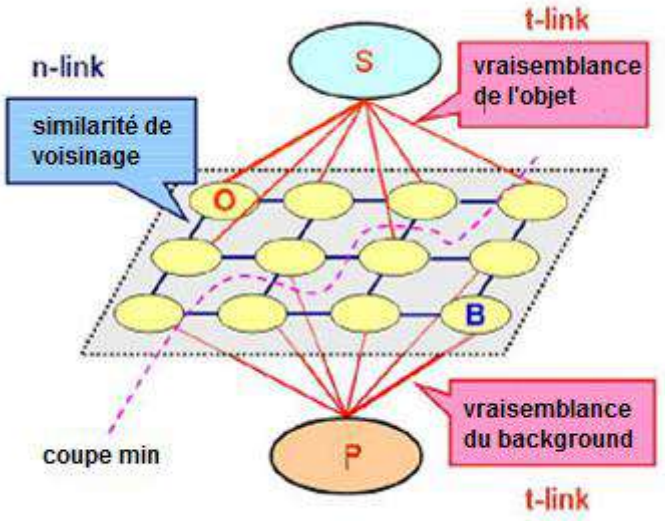


Figure 4.1 : Illustration d'une coupe de graphe sur une image de dimension 3x4 [FUKUDA08]

4.2.4 Calcul des capacités pour une configuration terminale:

En raison des contraintes de points de germes définis par l'utilisateur, l'attribution des poids initiaux aux arêtes est tributaire du point de germe affecté à la classe objet (″obj″) ou à la classe arrière plan (″bkg″) de l'image à segmenter. Cette configuration représente le terme "région" *R(A)*.

- Calcul de la capacité $P_{\text{classe_obj}}$ liée à la classe objet :

$$P_{classe_obj} = \begin{cases} K & \text{si } I \in I_{obj} \\ 0 & \text{si } I \in I_{bkg} \\ \lambda.\ P_{obj}(I) & \text{les autres cas} \end{cases} \qquad (4.3)$$

- Calcul de la capacité $P_{\text{classe_bkg}}$ liée à la classe arrière plan :

$$P_{classe_bkg} = \begin{cases} 0 & \text{si I} \in \text{I}_{\text{obj}} \\ K & \text{si I} \in \text{I}_{\text{bkg}} \\ \lambda.\ P_{bkg}(I) & \text{les autres cas} \end{cases} \tag{4.4}$$

$$avec \begin{cases} P_{obj}(I) = -ln(Pr(I/"obj")) \\ P_{bkg}(I) = -ln(Pr(I/"bkg")) \end{cases}$$

$$et \quad K = 1 + \max_{i \in S} \sum_{j:\{i,j\} \in \mathcal{V}} P_{\{i,j\}} \tag{4.5}$$

où I est l'intensité du pixel, *Pr(I/"obj")* et *Pr(I/"bkg")* sont des probabilités conditionnelles qui peuvent être déduites des histogrammes normalisés (Figure 4.2) de l'objet d'intérêt et de l'arrière plan de l'image.

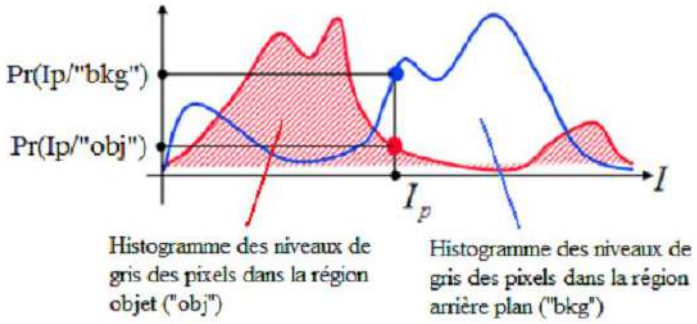


Figure 4.2 : Répartition des niveaux de gris des pixels suivant la classe objet et la classe arrière plan.

4.2.5 Calcul des capacités pour une configuration de voisinage :

La capacité des arêtes entre les nœuds du graphe est calculée à partir de la mesure de similarité P des pixels voisins en calculant le score fourni par :

$$P_{\{i,j\}} \propto exp\left(-\frac{(I_i - I_j)^2}{2\sigma^2}\right).\frac{1}{dist(i,j)} \tag{4.6}$$

où I_i (resp. I_j) est l'intensité du pixel positionnée en i (resp. j), dist(i,j) la distance Euclidienne entre les positions i et j et σ le paramètre qui contrôle l'affinité de teinte des pixels.

L'intérêt de cette mesure réside sur l'apport d'informations que nous fournissent le premier terme qui permet de discriminer les objets non semblables par la différence de teintes et le deuxième terme qui évalue les proches voisins des pixels par leur distance. Cette configuration de voisinage représente le terme de contour ou bord *B(A)*.

4.3 Implantation de la technique de segmentation :

4.3.1 Apprentissage :

A l'aide de la souris, l'utilisateur marque interactivement des pixels I_{obj} que nous considérons appartenir aux labels objet et d'autres pixels I_{bkg} que nous considérons appartenir aux labels fond ou arrière plan. Dans le cadre d'un processus de coupe de graphe, ces marquages jouent un rôle double : ils sont utilisés à la fois comme germe de segmentation et comme base d'apprentissage pour l'attribution des poids à l'énergie de vraisemblance via la construction des histogrammes décrits précédemment.

4.3.2 Segmentation d'image par minimisation d'énergie :

La procédure de segmentation envisagée ici consiste maintenant à reformuler les énergies de vraisemblance et d'a priori de la fonction objective (équation 4.1).

Nous calculons la couleur *C(i)* de la région *i* comme étant la moyenne des couleurs des régions présentant des similarités en termes de couleur. Afin d'extraire l'ensemble des arêtes, nous évaluons enfin le coefficient *C(i, j)* qui lie les contrastes de couleur entre deux régions *i* et *j*.

Ensuite, nous cherchons la coupe min correspondant au flot maximal. Pour cela, nous avons implémenté en premier lieu l'algorithme de base des chemins améliorant. Après plusieurs tests nous avons constaté que cette technique est très lente.

Pour obtenir une segmentation plus rapide de l'image, nous avons recouru à l'algorithme TSTAP (Two search trees augmenting path) [BOYKOV04] qui est actuellement l'outil le plus efficace pour départager le graphe en deux sous ensembles en adéquation avec les contraintes établies par les marqueurs.

4.3.3 Organigramme générale de la technique de segmentation implémentée :

La technique de segmentation développée dans ce mémoire est constituée de quatre étapes. La première est de lire le fichier image originale de taille (hauteur x largeur). Ensuite, de cette image on construit un graphe (création des arêtes où chaque nœud est relié à ses quatre voisins adjacents dans une grille de taille (hauteur x largeur). La troisième étape consiste à trouver la coupe min. A partir de cette coupe on aboutit à un étiquetage binaire qui correspond aux deux zones de l'image segmentée (objet et arrière-plan). La figure 4.3 montre un organigramme général de cette technique.

Nous avons développé un programme principal qui implémente cet organigramme en utilisant le langage Matlab. En plus, nous avons développé un programme appelé "maxflowmex" (de type mex) en langage C qui assure le lien avec une autre fonction écrite en langage C, pour le calcul du flot maximum en utilisant l'algorithme cité précédemment : TSTAP [BOYKOV04].

4.4 Résultats expérimentaux :

La segmentation vise à effectuer une partition de l'image, dans le cas d'images IRM, en séparant l'objet du fond. Nous présentons dans cette partie les résultats obtenus par l'algorithme de coupe de graphe qui utilise des contraintes rigides au moyen des points de germes sélectionnés par l'utilisateur (la zone en rouge sur les figures représente les germes de l'objet et la zone en bleu représente les germes du background).

La méthode est testée sur plusieurs images médicales afin de mieux cerner l'applicatif du résultat de la segmentation.

Nous montrons aux figures 4.4-4.5 une extraction d'un objet d'intérêt. Nous observons durant la phase d'apprentissage que les choix et les positions des points de germes influencent grandement les résultats de la segmentation. Cette dépendance peut être due à des erreurs de probabilités fournies par les histogrammes des points de germes qui ne sont modélisés que par deux gaussiennes.

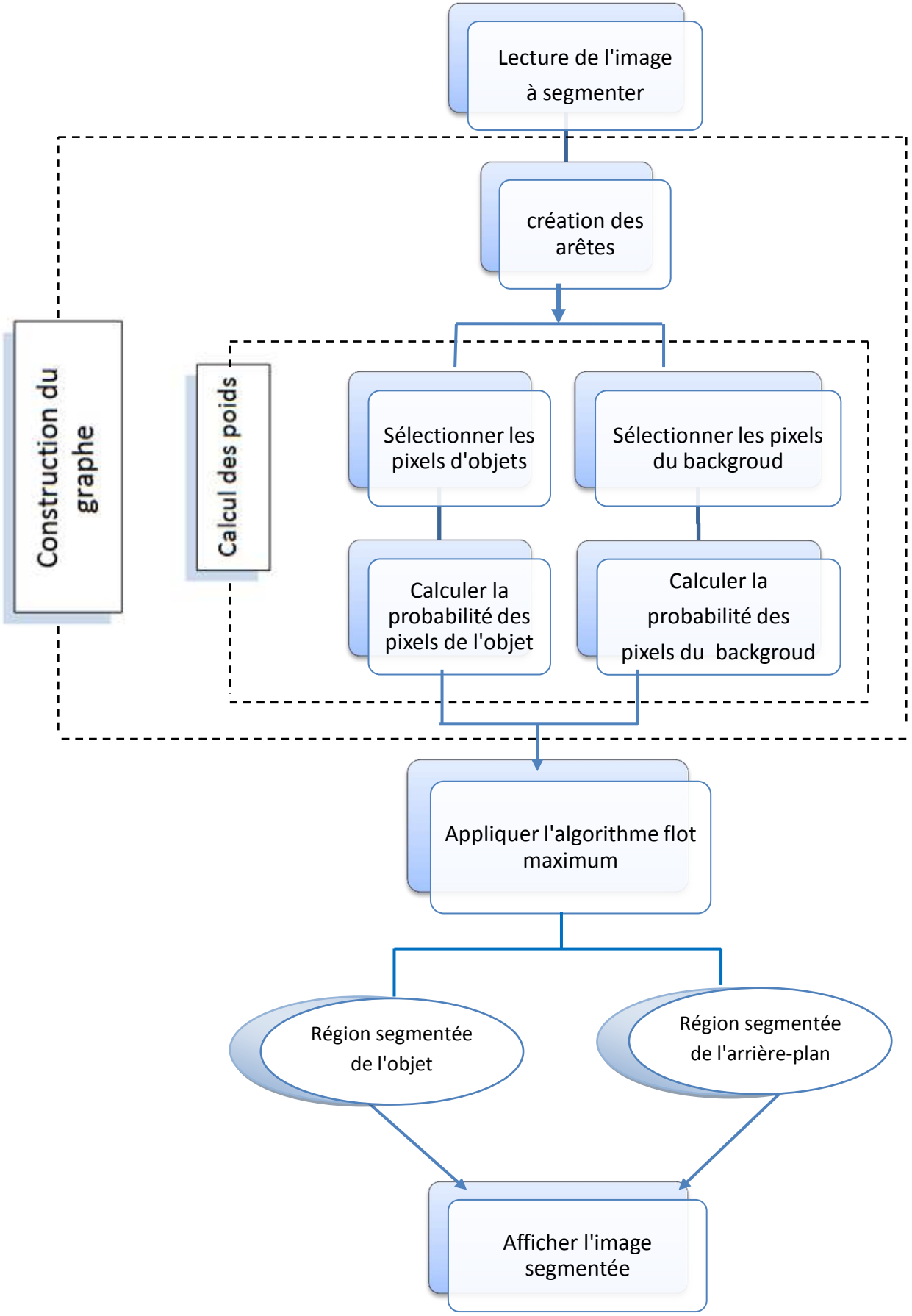
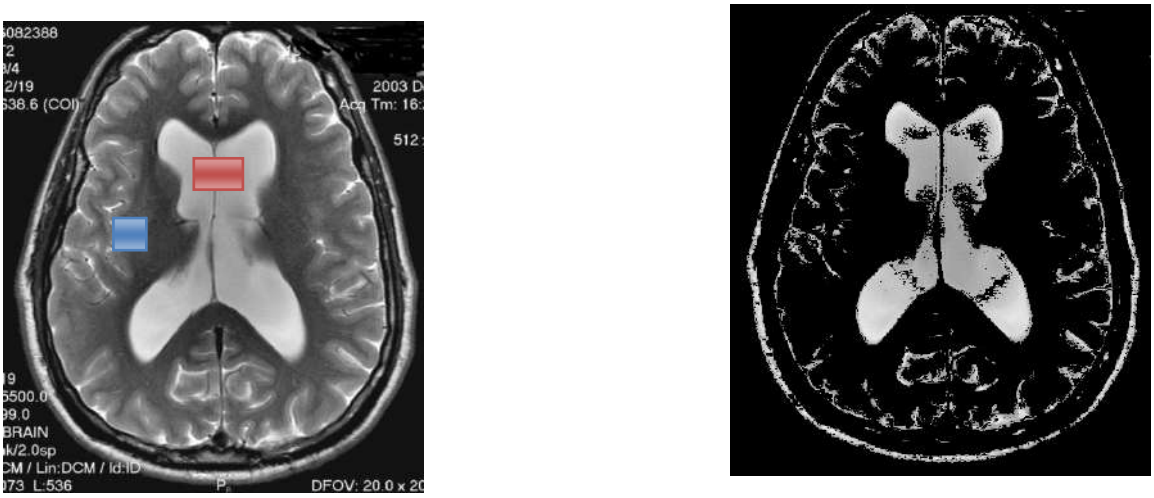
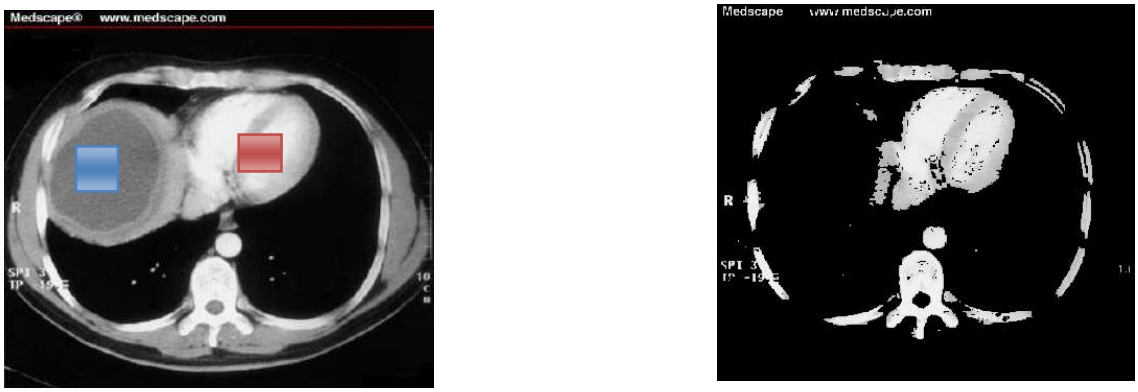


Figure 4.3 : Organigramme générale de la technique de segmentation implémentée



a) image originale b) image segmentée

Figure 4.4 : Segmentation d'une image IRM cérébrale



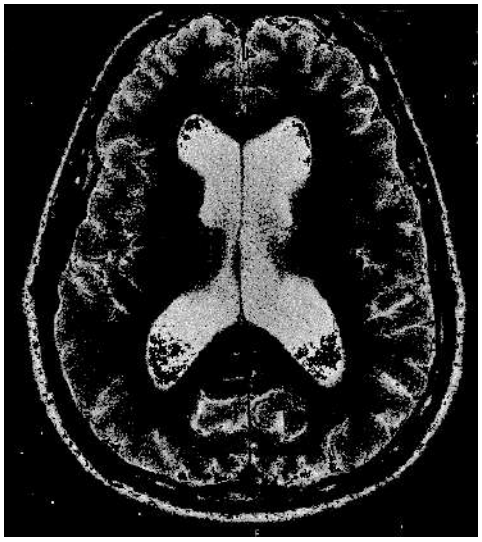
a) image originale b) image segmentée

Figure 4.5 : Segmentation d'une image IRM de l'abdomen

4.4.1 Effets du bruit sur la segmentation :

Pour tester la robustesse de l'algorithme, nous avons ajouté un bruit gaussien et utilisé les mêmes germes pour les deux images segmentées précédemment. L'effet du bruit peut être clairement vu dans l'image.

La partie image segmentée contient plus de l'image de fond que l'image précédente comme il est montré dans la figure 4.6.



(a)

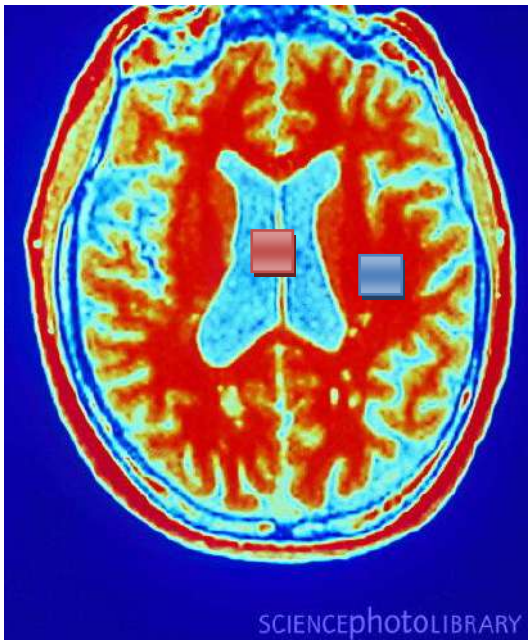


(b)

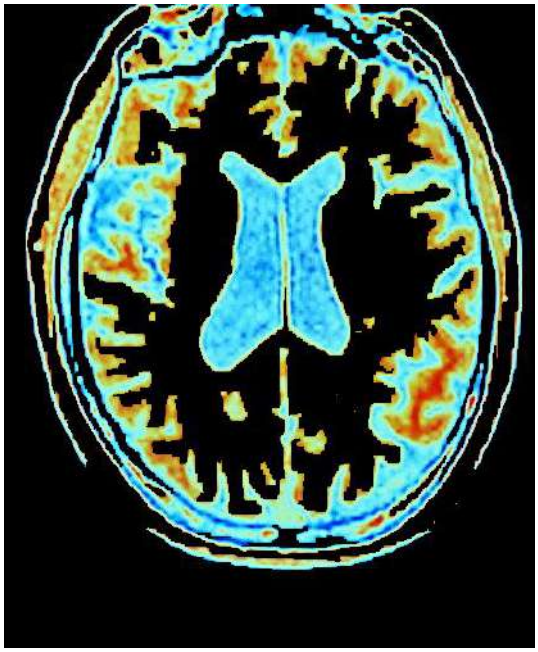
Figure 4.6 : Segmentation des images bruités : a) IRM cérébrale ; b) IRM de l'abdomen

4.4.2 Segmentation des images en couleurs :

Nous avons testé l'algorithme de segmentation implémenté sur des images médicales en couleurs. Les résultats de la segmentation sont représentés dans les figures 4.7 et 4.8.

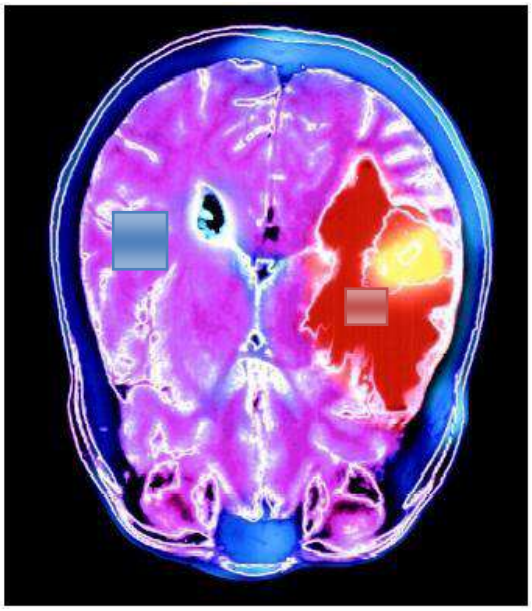


a) image originale

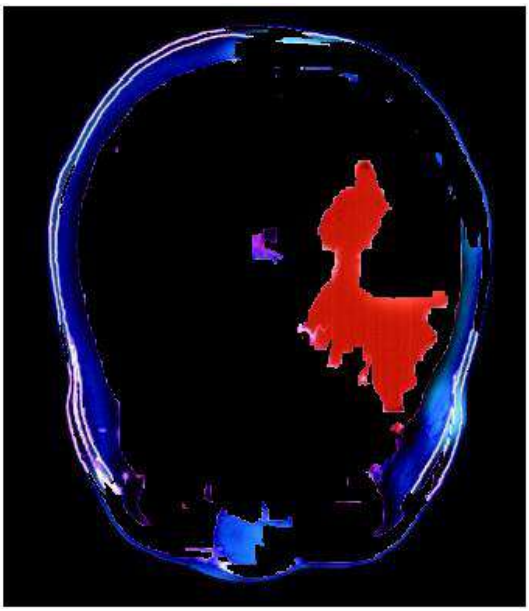


b) image segmentée

Figure 4.7 : Segmentation d'une image IRM cérébrale en couleur



a) image originale



b) image segmentée

Figure 4.8 : Segmentation de la région en rouge (tumeur) d'une image IRM cérébrale

Sur le plan pratique, les expériences sont menées sur un ordinateur portable muni d'un processeur Intel Core2Duo de 2,26GHz et d'une RAM de 4Go. Le temps de traitement pour segmenter les images de taille supérieure à 500x300 reste inférieur à 15 secondes. Ces résultats illustrent l'efficacité et la rapidité de la méthode pour extraire les objets d'intérêt de formes très variables. Pour ces résultats, nous avons fixé a priori les valeurs de λ=10 et de σ= 1.

Cependant, la segmentation présente encore des défauts superficiels de coupe qui se caractérisent par des contours relativement perturbés au niveau des régions texturées ou de faible contraste. Nous comptons néanmoins résoudre ce problème en recourant aux transformées en ondelettes qui, grâce à ses propriétés de filtrage, atténueront ces perturbations.

4.5 Conclusion :

Nous avons implémenté dans ce chapitre une méthode de segmentation utilisant l'algorithme de coupe de graphe contrainte par une connaissance a priori. Cette technique d'approche a montré des intérêts réels à la fois sur l'efficacité et la rapidité du rendu de la segmentation.

Quelques imprécisions de segmentation sont encore observées dans les régions de faible contraste et dans les zones texturées. Nous comptons dans les travaux futurs contourner ces problèmes et améliorer la qualité de la segmentation en faisant appel aux schémas de décomposition en ondelettes qui vont lisser l'image dans les basses fréquences du côté de la source *s* et atténuer les caractéristiques locales de texture en les filtrant dans les hautes fréquences du côté de puits *p*.

Conclusion Générale

Conclusion générale

Nous avons montré dans ce mémoire que le problème de segmentation d'images se ramènent à minimiser une fonction d'énergie que nous avons explicitée. Nous avons également mis en évidence que minimiser cette fonction d´énergie est équivalent à calculer la coupe minimale sur un graphe dont nous avons précisé les caractéristiques.

L'implantation d'un algorithme de graph-cuts nous a permis de tester cette technique et d'évaluer ces performances. Il apparait alors que l'emploi de la technique de graph-cuts en traitement d'images fournit une très bonne alternative aux heuristiques classiquement utilisées. On peut même envisager qu'elle supplante prochainement celles-ci car elle fait l'objet de divers travaux tendant à réduire sa complexité calculatoire et d'améliorer le minimum énergétique obtenu.

Cette méthode de segmentation semi-automatique rapide a été mise en place. Elle a été utilisée pour segmenter des images médicales. Elle ne nécessite qu'un court apprentissage, un degré d'interactivité relativement faible et est suffisamment rapide pour être utilisée dans un contexte médical.

Ce travail a permis de mettre en évidence les avantages d'une telle technique mais aussi ces limites dans le cadre de la segmentation. Nous proposons une résolution multi-échelle qui peut constituer une nette amélioration des résultats.

Elle consisterait à adapter la coupe de graphe aux schémas de décomposition en ondelettes qui, par effet de décimation, permet de comprimer et traiter une image de taille réduite. Nous espérons tout simplement que cette opération de sous échantillonnage ne dégrade pas trop les informations pixelliques locales et les frontières des objets à discriminer. Nous restons toutefois prudents sur celle-ci car il est difficile de satisfaire son critère de validité.

Quand la forme de l'objet est connue et que les contours de l'objet sont mal définis, le processus de segmentation peut être efficacement guidé par un a priori. Comme un deuxième perspective, nous proposons une segmentation par coupe de graphe, s'appuyant sur un a priori de forme obtenu par analyse en composantes principales en introduisant dans la fonction d'énergie un terme supplémentaire prenant en compte cet a priori.

Par la suite, l'objectif est d'affiner la segmentation. On remarque que la segmentation est imprécise lorsque les pixels de l'objet se rapprochent trop de ceux du fond. Un pixel de l'objet est facilement absorbée par ses voisins appartenant au fond. Or, il est assez fréquent d'avoir un pixel de l'objet entouré fortement de pixels du fond. On pourrait alors prendre en compte ce genre de cas en modifiant le poids des capacités sur les arcs (par exemple, en acceptant des valeurs signées sur les arcs). Ainsi, un pixel de l'objet au milieu du fond est difficilement accepté dans le fond alors qu'un pixel plus foncé au milieu des pixels plus clairs peut être considéré appartenant à l'objet.

D'autre part, une autre idée serait d'augmenter la connexité du graphe sur le quel on travaille et passer d'un graphe de 4-connexe à 8-connexe. Et finalement, nous procurons l'application de cette méthode dans le cas des images médicales en 3D avec des graphes de 6-connexe jusqu'à 26-connexe.

Bibliographie

[ABEND65] Abend, K., Harley, T.J., andKanal, L.N. Classification of binary random pattern. IEEE Transactions on Information Theory, 11, 538–544, 1965.

[ASLAN09] M. Aslan, A. Ali, H. Rara, B. Arnold, A. Farag, R. Fahmi et P. Xiang : A novel 3D segmentation of vertebral bones from volumetric CT images using graph cuts, in Advances in Visual Computing, vol. 5876, 2009, pp. 519–528.

[BEINEK68] Lowell W. Beineke: On derived graphs and digraphs, pages 17–33. Teubner-Verlag, 1968.

[BERGE62] Claude Berge: Theory of graphs. Methuen and Co., 1962.

[BESAG74] Besag, J. : Spatial interaction and the statistical analysis of lattice systems.
J. Royal Statist. Soc. (series B), 36, 192-326,1974.

[BESAG86] Besag, J. : On the statistical analysis of dirty pictures. J. R. Statist. Soc. B, 48(3), 259-302, 1986.

[BOYKOV98] Yuri Boykov, Olga Veksler et Ramin Zabih: Markov random fields with efficient approximations. In Proceedings of the1998 IEEE Computer Society conference on Computer Vision and Pattern Recognition, pages 648–655. IEEE Computer Society, 1998.

[BOYKOV99] Yuri Boykov, Olga Veksler et Ramin Zabih: A new algorithm for energy minimization with discontinuities. In Proceedings of the 2nd International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition, pages 205–220, London, UK, 1999. Springer-Verlag .

[BOYKOV00] Y. Boykov et M. P. Jolly : Interactive organ segmentation using graph cuts, MICCAI, pp. 276–286, 2000.

[BOYKOV01] Y. Boykov, M. P. Jolly: Interactive graph cuts for optimal boundary and region segmentation, in ICCV, volume I, pp. 105-112, 2001.

[BOYKOV01a] Yuri Boykov, Olga Veksler et Ramin Zabih: Fast approximate energy minimization via graph cuts. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(11): 1222–1239, 2001.

[BOYKOV04] Yuri Boykov et Vladimir Kolmogorov: An experimental comparison of mincut/max-flow algorithms for energy minimization in vision. IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(9):1124–1137, 2004.

[COHEN91] L. D. Cohen et I. Cohen : Finite element methods for active contour models and balloons for 2D and 3D images, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 15, pp. 1131–1147, 1991.

[COOK98] William J. Cook, William H. Cunningham, William R. Pulleyblank et Alexander Schrijver: Combinatorialoptimization. JohnWiley &Sons, Inc.,New York,NY, USA, 1998.

[CORMEN01] Thomas H. Cormen, Charles E. Leiserson et Ronald L. Rivest : Introduction to algorithms, second edition. MIT Press, 2001.

[DANTZIG56] Georges Bernard Dantzig et Delbert Ray Fulkerson: On the max-flow min-cut theorem of networks. In Linear inequalities and related systems, volume 38 de Annals of Mathematics Studies, chapitre 17, pages 225–231. Princeton University Press, Princeton, 1956.

[DEGIOR95] Daniele Giorgio Degiorgi et Klaus Simon: A dynamic algorithm for line graph recognition. In Proceedings of the 21 st International Workshop on Graph-Theoretic Concepts in Computer Science, pages37–48, London, UK, 1995. Springer-Verlag.

[DESCOM93] Descombes, X. : Champs Markoviens en analyse d'images. Ph.D. thesis, Ecole Nationale Supérieure des Télécommunications, 1993.

[DIESTEL97] Reinhard Diestel: Graph theory. Graduate Texts in Mathematics. Springer, 1997.

[DINIC70] E. A. Dinic: Algorithm for solution of a problem of maximum flow in networks with power estimation. Soviet Mathematics Doklady, 11:1277–1280, 1970.

[EDMONDS72] Jack Edmonds et Richard Manning Karp: Theoretical improvements in algorithmic efficiency for network flow problems. Journal of the Association for Computing Machinery, 19(2):248–264, April1972.

[FORD62] Lester Randolph Sr Ford et Delbert Ray Fulkerson: Flows in networks. Princeton University Press, 1962.

[FREED05] Daniel Freedman et Petros Drineas: Energy minimization via graph cuts: settling what is possible. In Proceedings of the2005 IEEE Computer Society conference on Computer Vision and Pattern Recognition, pages 939–946, Washington, DC, USA, 2005. IEEE Computer Society.

[FUKUDA08] K. Fukuda, T. Takiguchi, Y. Ariki, Graph cuts using local texture features of wavelet coefficient for image segmentation, IEEE International Conference on Multimedia and Expo, June 23-26, 2008, Hannovre, Germany.

[GARCIA09] D. Garcia-Lorenzo, J. Lecoeur, D. L. Arnold, D. L. Collins et C. Barillot : Multiple sclerosis lesion segmentation using an automatic multimodal graph cuts, Med Image Comput Assist Interv, vol. 12, pp. 584–591, 2009.

[GEMAN84] Geman, S. and Geman, D. : Stochastic Relaxation, Gibbs Distribution, and the Bayesian Restauration of Images. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-6(6), 721-741, 1984.

[GEMAN85] Geman, S. and McClure, D. E. : Bayesian image analysis: an application to single photon emission tomography. Proc. Statist. Comput. sect. (Amer. Statist. Assoc. Washington DC), pages 12-18, 1985.

[GOLDBERG98] Andrew V. Goldberg, Eva Tardos et Robert Endre Tarjan: Network flow algorithm, Paths, flows and VLSI-design, pages 101–164, 1998.

[GREIG89] D. M. Greig, B. T. Porteous et A. H. Seheult: Exact maximum a posteriori estimation for binary images. Journal of the Royal Statistical Society, 51(2):271–279, 1989.

[KANAL80] Kanal,L.N. : Markov mesh models. Image Modeling, New York, Academic Press, 1980.

[KARP72] Richard Manning Karp: Reducibility among combinatorial problems. In complexity of Computer Computations, pages 85–103. Plenum Press, 1972.

[KASS88] M. Kass, A. Witkin, et D. Terzopoulos : Snakes : Active contour models, International Journal of ComputerVision, vol.1, pp.321–332,January1988.

[KIRKPA82] Kirkpatrick, S., Gellatt, C. D., and Vecchi, M. P. : Optimization by simulated annealing. IBM Thomas J. Watson research Center, Yorktown Heights,NY, 1982.

[KOLMO02] Vladimir Kolmogorov et Ramin Zabih: What energy functions can be minimized via graph cuts? In Proceedings of the 7th European Conference on Computer Vision, pages 65–81. Springer-Verlag, 2002.

[KOLMO05] V. Kolmogorov et Y.Boykov : What metrics can be approximated by geo-cuts, or global optimization of length area and flux, International Conference on Computer Vision (ICCV), Beijing, China, vol. 1, pp. 564–571, 2005.

[LEHOT74] Philippe G. H. Lehot: An optimal algorithm to detect a line graph and output its root graph. Journal of the Association for Computing Machinery, 21(4):569–575, 1974.

[LI04] Y. Li, J. Sun, C.K. Tang, H. Y. Shum, Lazy Snapping, ACM Transactions on Graphics, vol. 23, pp. 303-308, 2004.

[OSHER03] S. Osher et N. Paragios : Geometric Level Set Methods in Imaging, Vision and Graphics. Springer, 2003.

[PUNDLIK10] S. J. Pundlik, D. L. Woodard et S. Birchfield : Iris segmentation in non-ideal images using graph cuts, Image Vision Comput., pp.1671–1681, 2010.

[ROUSSO73] Nicholas D. Roussopoulos: A max {m, n} algorithm for determining the graph H from its line graph C. Information Processing Lett, 2(4):108–112,1973.

[SCHRIJVER03] Alexander Schrijver: Combinatorial optimization: polyhedra and efficiency, volume A. Springer, 2003.

[SETHIAN99] J. A. Sethian : Level Set Methods and Fast Marching Methods. Cambridge University Press, 1999.

[SHI00] J. Shi, J. Malik : Normalized cuts and image segmentation, IEEE PAMI, 22(8), pp. 888-905, August 2000.

[WHITN32] Hassler Whitney: Congruent graphs and the connectivity of graphs. American Journal of Mathematics, 54:150–168, 1932.

[WU82] Wu, F. Y. : The potts model. Reviews of Modern Physics, 54(1), 235-267, 1982.