Paradigmes de segmentation de graphe : comparaisons et applications en traitement d'images

Cédric Allène

Université Paris-Est
École doctorale ICMS

Thèse

présentée en vue d'obtenir le grade de Docteur,
spécialité *Informatique*,

par

Cédric Allène

Paradigmes de segmentation de graphe : comparaisons et applications en traitement d'images

Thèse soutenue le 12 Février 2009 devant le jury composé de :

M.	Fernand Meyer	Mines ParisTech - CMM	(Président)
M.	Laurent Cohen	Université Paris IX Dauphine - CEREMADE	(Rapporteur)
M.	Philippe Salembier	Universitat Politècnica de Catalunya - STCD	(Rapporteur)
M.	Nikos Paragios	École Centrale de Paris - MAS	(Examinateur)
M.	Michel Couprie	ESIEE - A^2SI	(Directeur)
M.	Renaud Keriven	École des Ponts ParisTech - CERTIS	(Directeur)

À mes Parents…
À ma Schoubidounette…

Remerciements

Dire merci... Cela peut sembler une tâche tellement simple qui, au final, peut se révéler terriblement plus ardue que prévu...

En effet, derniers paragraphes à rédiger pour parachever ce mémoire, la section des remerciements n'en est pas pour autant la plus aisée à écrire à mon goût (même si certains chapitres m'ont déjà donné bien du mal) ! Il faut être certain de n'omettre personne tout en restituant à leur plus juste valeur les apports de chacun, aussi bien sous forme de contributions directes qu'en termes de soutien, afin d'exprimer au mieux la gratitude que l'on ressent vis-à-vis de toutes les personnes avec qui l'on a collaboré, de près ou de loin, ou qui nous ont épaulé, d'une façon ou d'une autre, au cours de ces quatre années de doctorat et ayant permis son aboutissement.

Malgré les remarques précédentes, je commencerai mes remerciements d'une manière un peu inhabituelle...

La rédaction de ce mémoire en tant que simple exercice de style servant à valider mes travaux de recherches effectués durant mon doctorat me paraissait une tâche des plus rébarbatives à laquelle j'avais du mal à m'atteler. L'idée de produire un document pédagogique regroupant mes différents travaux qui soit accessible à un néophyte dans le domaine tout en offrant des réponses précises aux questions qu'une personne plus chevronnée pourrait se poser sur un des thèmes qui y sont exposés était une bien plus grande source de motivation en dépit de la charge de travail supplémentaire que cela représentait. C'est dans cette optique que j'ai rédigé ce manuscrit comme une synthèse des documents que j'aurais souhaité trouver durant mon doctorat pour répondre aux interrogations que je pouvais avoir, allant du simple détail technique peu détaillé dans la littérature au lien théorique entre méthodes de segmentation mis en évidence durant cette thèse. Bien sûr, tout cela a été formulé selon ma propre vision des choses qui peut très bien ne pas convenir à tout le monde, mais qui, je l'espère, s'avérera adaptée au plus grand nombre.

C'est pourquoi je remercie toute personne prêtant attention à ce mémoire en ayant le courage de se plonger dans quelques unes des 300 pages le constituant afin d'en retirer quelque chose, que ce soit la découverte d'un nouveau domaine de recherche ou la réponse à une question précise, car, si elle y arrive, alors cet ouvrage aura atteint son objectif à mes yeux !

Revenons maintenant à des remerciements plus conventionnels s'adressant à des personnes que je connais bel et bien !

En premier lieu, je souhaiterais exprimer toute ma reconnaissance à mes directeurs de thèse pour leur encadrement, le temps qu'ils m'ont consacré, les nombreuses discussions échangées, les réponses qu'ils m'ont apportées et les questions qu'ils ont su me faire sou-

lever, sans oublier leur bonne humeur habituelle qui permettait d'aller de l'avant lorsque les résultats escomptés n'étaient pas au rendez-vous.

Un grand merci, donc, à Michel Couprie pour tout ce qu'il m'a appris sur les graphes, ses conseils de rédaction, ses relectures minutieuses et, surtout, pour m'avoir introduit dans le monde de la recherche via un stage en traitement d'imagerie médicale effectué au laboratoire A^2SI de l'ESIEE en 2003 (co-encadré avec Laurent Najman), me donnant ainsi l'envie de poursuivre dans cette voie.

Merci également à Renaud Keriven pour m'avoir accueilli au sein du CERTIS, fait découvrir différentes facettes de la vision par ordinateur, accordé le financement nécessaire à la bonne terminaison de ma thèse et permis de découvrir Hong-Kong (Chine), Rio de Janeiro (Brésil) et Tampa (Floride) grâce aux conférences où il m'a envoyé.

Être encadré par deux directeurs de thèse d'horizons et de savoir-faires différents m'a ainsi donné l'occasion de profiter du "meilleur des deux mondes". Cette thèse n'aurait pu être réalisée sans leur aide à tous les deux.

Je tiens également à remercier les autres professeurs du CERTIS avec qui j'ai eu l'opportunité de collaborer.

Tout d'abord, merci à Nikos Paragios, suiveur de mon stage de DEA (appelé *Master 2* désormais) et prédécesseur de Renaud Keriven dans l'encadrement de mon début de thèse au CERTIS avant de nous quitter pour rejoindre le laboratoire MAS de l'École Centrale de Paris, pour m'avoir permis de rédiger mon premier article et pour son encadrement impliqué, restant jusqu'à des heures tardives, même le week-end, pour aider ses thésards à soumettre leur article quelques heures, voir quelques minutes, avant une *deadline*.

Merci à Jean-Yves Audibert pour m'avoir apporté son aide précieuse sur la rédaction de la preuve du théorème 5.20 (publié dans [7, 8]), alors que les graphes n'étaient pas son domaine de prédilection (sans doute a-t-il eu pitié de moi lorsqu'il me voyait m'arracher les cheveux à m'acharner dessus) !

Enfin, merci à Jean-Philippe Pons pour ses connaissances et ses programmes en reconstruction 3D à partir de vues multiples dont il m'a fait profiter ainsi que pour ses nombreuses bonnes idées, son optimisme sans borne et sa permanente jovialité !

Tous mes remerciements aux membres de mon jury de soutenance de thèse pour avoir fait le déplacement (de loin pour certains) afin d'assister à l'exposé de mes travaux et, plus particulièrement, à Fernand Meyer, pour avoir présidé mon jury de soutenance de thèse, ainsi qu'à Laurent Cohen et Philippe Salembier pour avoir accepté d'être rapporteur de ce manuscrit malgré la charge de travail que cela représente.

De manière générale, je tiens à remercier chaleureusement tous les professeurs, doctorants, post-doctorants, stagiaires ou autres qui ont contribué à faire de ma thèse une expérience aussi enrichissante au travers de nos échanges scientifiques fructueux, bien sûr, mais également amicaux !

Merci aux membres du CERTIS d'avoir enduré mes suggestions incessantes de pots pour tout motif possible et imaginable avec autant d'entrain et, surtout, d'avoir été autant à accepter de jouer le jeu, permettant de tous se retrouver régulièrement autour d'un goûter ou autre. Ces moments de convivialité vont beaucoup me manquer ! Bien sûr, cette ambiance sympathique régnant au CERTIS n'aurait été possible sans la participation motivée de certains :

– Anne-Laure Jachiet (ou plutôt, Mme Chauve, depuis peu !), une colocataire de bureau et une cuisinière hors pair à la bonne humeur communicative, toujours prête à rendre service, ma référence en ce qui concerne LaTeX, mais qui a trop souvent commis l'erreur stratégique (à cause de sa trop grande gentillesse) de me proposer de

partager ses différents encas (ce que je ne pouvais refuser...) et dont j'ai dû ébranler la patience à de nombreuses reprises lorsqu'elle m'attendait pour prendre le chemin du départ alors que j'avais systématiquement un "petit truc" à finir avant de partir.
- Anne-Marie Tousch, toujours souriante, même quand elle râle après ses élèves et dont l'aptitude à rapporter régulièrement de succulentes tablettes de chocolat suisse aura profité à plus d'un (moi le premier !) ;
- Brigitte Mondou, secrétaire du CERTIS toujours prête à lutter contre administrations ou entreprises pour nous aider dans nos démarches (telles que la planification des missions, par exemple) et subvenir à notre confort ;
- Vu Hoang Hiep, dont la faculté à empêcher un PC de planter juste en restant à côté (dans une certaine limite tout de même) me surprendra toujours, mais peut-être pas autant que les parfums des gelées vietnamiennes dont il nous a fait profiter ;
- Ehsan Aganj, dernier pianiste émérite à demeurer encore au CERTIS (mais plus que pour quelques mois encore) ;
- Nicolas Thorstensen, il dit très souvent avoir faim mais, finalement, n'est pas un concurrent trop rude durant les pots ;
- Pascal Monasse et Arnak Dalalyan, les deux derniers permanents arrivés au CERTIS et qui se sont bien adaptés à ses coutumes (Arnak nous ayant gratifié d'un succulent repas arménien pour fêter ses 30 ans).

Je n'en oublie pas pour autant les "anciens" du CERTIS qui en sont partis avant moi dont, notamment :
- Alexandre Chariot, mon compagnon d'infortune dans l'administration et la gestion du parc informatique (peu ont, comme nous, connu la joie de la planification des sauvegardes et de la recherche des causes de leurs plantages) et qui, bien que plus discret que moi, était souvent un sérieux rival pour prendre la dernière part de gâteau ;
- Jaonary Rabarisoa, notre photographe passionné qui a attrappé ce virus pendant qu'il était au CERTIS et a su le transmettre à plusieurs d'entre nous (avec des symptômes peut-être amoindris cela dit) ;
- Lokman Abbas-Turki et Patrick Etyngier, qui faisaient régulièrement le concours de celui qui arriverait le plus tôt au labo ;
- Zsolt Janko, post-doctorant hongrois qui a voulu nous initier aux alcools de chez lui (et avec les mêmes quantités que chez lui également) ;
- Thomas Ailleret, qui va toujours "mieux que jamais" et partage mon avis sur le statut des "Colons de Catane" comme étant le meilleur jeu de plateau au monde ;
- Victor Nicollet, dont la capacité à ingurgiter d'affilée des pastilles Vichy est phénoménale ;
- Maxime Taron, mélomane, fin gourmet et œnologue en devenir qui fut mon colocataire de bureau lors de sa visite hebdomadaire, toujours bienvenue, alors qu'il avait "officiellement" déménagé à l'ECP ;
- Romain Dupont, également un colocataire de bureau pendant un temps, toujours joyeux, on ne s'ennuie jamais avec lui, il nous a animé de nombreuses pauses et déjeuners grâce à sa philosophie improbable et ses théories alambiquées et pas toujours bien étayées, sa technique, dite du "bulldozer", au basket ball restera dans les mémoires ;
- Noura Azzabou, qui a partagé avec moi de grands moments d'incertitude lors de notre étude des filtres à particule ;
- Charlotte Ghys et Olivier Juan, le premier (et unique, à ma connaissance) couple que le CERTIS ait permis de faire se rencontrer, réunissant ainsi une pro de l'aiguille à tricoter et un véritable *geek* parmi les *geeks* ;
- Geoffray Adde, notre coach sportif des débuts du CERTIS qui a réussi à y monter

une équipe de basket ball et essayé, tant bien que mal, de m'enseigner quelques rudiments de ce sport (avec plus ou moins de finesse dans mon exécution - Jean-Yves se souvient sûrement encore de la fois où je m'initiais à la "passe éclair" !), ce qui nous a permis de passer de très bons moments ensemble sur le terrain ;
– Fabien Le Jeune, le seul parmi toutes les personnes passées au CERTIS qui ait partagé mon goût (ou du moins, le seul qui l'ait avoué) pour les vieux films de série B ou Z ainsi que certaines séries du même genre.

Merci à l'ensemble du corps enseignant de l'ESIEE avec lequel j'ai collaboré durant mon monitorat là-bas. Cette expérience fut des plus instructives même si certaines heures de travaux pratiques sur ordinateur ressemblaient davantage à de la garderie... Cela dit, j'ai tout de même eu le plaisir de voir quelques élèves s'orienter vers la recherche, notamment John Chaussard, devenu actuellement thésard au laboratoire A^2SI et avec qui j'ai toujours autant de plaisir à discuter (de sujets plus ou moins proches de la thèse - plutôt moins, je dois le reconnaître).

J'en profite également pour remercier Olena Tankyevych et Hugues Talbot, respectivement doctorante et professeur au laboratoire A^2SI, avec qui j'ai eu la chance de faire davantage connaissance lors de notre séjour à Rio de Janeiro (Brésil) grâce à nos pérégrinations communes (qui nous ont values quelques petites mésaventures agrémentées de sueurs froides - je me souviendrai toujours de mon petit sprint face aux coutumes peu accueillantes de trois jeunes *Cariocas* qui semblaient très intéressés par mon sac à dos).

Merci à Sylvie Cach, chargée des inscriptions et du suivi des thésards de l'Université Paris-Est, pour son aide et sa patience envers moi dans les démarches administratives requises pour ma soutenance de thèse.

En cette fin de doctorat, j'ai également une pensée pour Jorge Cham, auteur des fameux *PhD Comics* (voir un exemple dans la figure 1), que je remercie pour nous divertir si souvent avec ses illustrations et ses histoires sur le monde de la thèse qui, si certaines peuvent s'avérer mystérieuses en début de doctorat, se révèlent parfois bien proches de la vérité lorsque l'on a acquis une ou deux années d'expérience en tant que thésard...

Un merci tout particulier aux membres de ma famille (j'y inclus, bien évidemment, ma belle-famille) et aux amis proches qui se sont enquis régulièrement de mon avancement et de mon état moral au cours de ces années malgré mon manque de temps flagrant pour les voir aussi souvent que j'aurais souhaité (spécialement pour la branche bretonne de ma famille qui ne m'aura, je crois, vu qu'une seule fois durant mon doctorat). Leurs encouragements et leur soutien m'ont été précieux, surtout durant cette dernière ligne droite (beaucoup plus longue que je ne l'avais imaginée) que fut celle de la rédaction de ce manuscrit. Quoi qu'il en soit, ça y est, je peux leur annoncer que ma thèse est "enfin" achevée !

Une mention spéciale à mes amis *"Trolls"* de Taverny (incluant, bien entendu, ceux qui se sont expatriés) et, plus particulièrement, Charlie Chagny, Florian Carnini et Guillaume Parodi, qui ont eu le courage de venir assister à ma soutenance ! Je reconnais les avoir bien délaissés durant tout ce temps, passant d'une rencontre quasi-hebdomadaire en début de thèse à une visite semestrielle ces deux dernières années... J'espère que nous aurons l'occasion de rattraper le temps perdu !

Me rendant compte que mes remerciements sont autrement plus longs que je ne l'imaginais initialement, je ne me risquerai pas à entrer dans une nouvelle liste qui risquerait d'être trop longue... Je suis persuadé que, malgré qu'ils ne soient pas cités explicitement, ces proches, pour qui je ne manque pas d'avoir une pensée, se reconnaîtront aisément !

Ayant pour habitude de garder le meilleur pour la fin, je ne saurais conclure autrement ces remerciements qu'en essayant (peut-être en vain) de trouver les mots pour exprimer mon immense gratitude aux personnes qui me tiennent le plus à cœur (et à qui je dois un grand merci spécial supplémentaire pour le splendide "jouet" qu'ils m'ont offert en guise de récompense de fin de thèse : un appareil photo reflex Nikon D60).

Mes Parents tout d'abord, Monique et Gilbert Allène, que je ne saurais jamais remercier assez pour tout ce qu'ils ont fait pour moi : outre l'éducation et les études qu'ils m'ont offerts, même s'ils disent aujourd'hui être dépassés par mon travail, c'est à eux que je dois mon intérêt pour les sciences et, en particulier, à mon Papa qui m'a transmis son goût prononcé pour les ordinateurs ! Je suis bien conscient de la chance que j'ai de les avoir comme parents car je ne serais probablement jamais arrivé où j'en suis sans eux et leur soutien indéfectible. Je vous aime fort et vous remercie infiniment pour tout ce que vous m'avez apporté !

Valérie Bourgogne, ma Schoubidounette, la femme que j'aime à la folie et qui rend ma vie plus belle depuis presque quatre années déjà. Elle a enduré ma thèse avec une grande patience, surtout durant la rédaction de ce mémoire, pendant laquelle, bien que nous habitions ensemble, elle ne m'a pas beaucoup vu, se retrouvant avec l'ensemble des tâches ménagères à gérer et un conjoint au moral parfois chancelant devant la charge de travail restant à accomplir. Sa présence à mes côtés et son Amour furent (et resteront) les meilleurs des remontants. Je t'adresse mes plus tendres remerciements accompagnés de tout mon Amour !

Champs-sur-Marne, Avril 2009.

FIGURE 1 – *PhD Comics* parus en Avril 2009, copyright Jorge Cham (www.phdcomics.com)

Titre:
Paradigmes de segmentation de graphe : comparaisons et applications en traitement d'images

Résumé:
Les techniques de segmentation de graphe sont souvent utilisées en traitement d'images puisque ces dernières peuvent être vues comme des graphes valués.
Dans cette thèse, nous montrons des liens existant entre plusieurs paradigmes de segmentation de graphes valués. Nous présentons tout d'abord différentes définitions de ligne de partage des eaux et sélectionnons celle dont le cadre permet la comparaison avec des forêts couvrantes particulières. Nous montrons qu'une telle ligne de partage des eaux relative à des marqueurs arbitraires est équivalente à une coupe induite par une forêt couvrante de chemins de moindre altitude. Ensuite, les coupes induites par des forêts couvrantes de poids minimum sont démontrées comme étant des cas particuliers ayant l'avantage d'éviter certaines segmentations non souhaitées. Enfin, nous montrons qu'une coupe minimale coïncide avec une coupe induite par une forêt couvrante de poids maximum pour certaines fonctions de poids particulières.
Dans une seconde partie, nous présentons deux applications utilisant la segmentation de graphe : la renaissance d'images et le mélange de textures pour la reconstruction 3D.

Mots-clés:
Graphe, segmentation, ligne de partage des eaux, forêt couvrante, coupe minimale, renaissance d'image, mélange de textures

Title:
Graph segmentation paradigms : comparisons and applications in image processing

Abstract:
Graph segmentation techniques are often used in image processing since an image can be seen as a weighted graph.
In this thesis, we show some links existing between several weighted graph segmentation paradigms. We first present different definitions of watersheds and select the one which framework allows comparison with specific spanning forests. We show that such a watershed relative to arbitrary markers is equivalent to a cut induced by a shortest path spanning forest. Then, cuts induced by minimum spanning forests are demonstrated as being particular cases which advantageously avoid some undesirable results. Finally, we show that minimum cuts coincide with cuts induced by maximum spanning forests for some particular weight functions.
In a second part, we present two applications using graph segmentation : image renaissance and texture blending for 3D reconstruction.

Keywords:
Graph, segmentation, watershed, spanning forest, min-cut, image renaissance, texture blending

Table des matières

Liste des figures

Introduction

Une image vaut mille mots.

Confucius (*Kongfuzi* en chinois, -551 à -479 av. J.-C.)

La vue est, pour beaucoup, le plus important des cinq sens que possède l'être humain. Et pour cause, 80% des informations sensorielles perçues par notre cerveau proviennent de la vision. La rétine en est l'élément d'acquisition puisqu'elle est couverte de photorécepteurs : quelques 130 millions de bâtonnets et 65 millions de cônes. Les bâtonnets sont très sensibles à l'intensité lumineuse alors que les cônes le sont aux couleurs. En 2006, une équipe de chercheurs de l'*University of Pennsylvania School of Medicine* exposa dans le numéro de Juillet de la revue *Current Biology* que le débit d'informations entre la rétine et le cerveau était équivalent à une connexion Ethernet de 10Mbit/s.

Contrairement à ce que l'on pourrait s'attendre, 90% des informations qui transitent dans les nerfs optiques vont du cerveau vers les yeux et non l'inverse. L'œil est donc loin d'être un capteur optique passif : avec l'aide du cerveau, il cherche à interpréter l'information qu'il reçoit selon des schémas préalablement appris. Ainsi, il est très facile de reconnaître le visage de l'un de ses proches sur une photo par exemple. Par contre, si on retourne celle-ci, cela devient beaucoup plus dur. Cela est tout simplement dû au fait que notre cerveau n'est pas entraîné à le discerner dans cette position. Il en résulte que la vue est le sens le plus susceptible d'être leurré puisqu'il cherche à appliquer les images mentales qu'il connaît sur les informations qu'il perçoit, quitte à en imaginer certaines là où elles n'y sont pas, créant ainsi ce qu'on appelle des illusions d'optique.

Avec de telles aptitudes, il n'est pas surprenant que la place qu'occupe l'image, sous toutes ses formes, soit si importante dans notre société actuelle : peinture, photographie, films, jeux vidéo, imagerie médicale, assistance au diagnostic, météorologie, cartographie, vidéo surveillance, contrôle de qualité, tourisme, publicité,...

Le contenu d'une image peut être très dense en informations. Notre cerveau arrive, la plupart du temps, à les dissocier en reconnaissant les différents éléments représentés. Cependant, si nous arrivons à faire cela sans peine après des années d'apprentissage durant la petite enfance, il est des plus difficile de reproduire ce phénomène artificiellement.

La vision artificielle, aussi appelée vision par ordinateur ou, plus récemment, vision cognitive, est un problème de traitement et d'analyse d'images correspondant à une branche de l'intelligence artificielle. Celle-ci a pour but de faire "comprendre" à une machine ce qu'elle "voit" (à travers un système d'un ou plusieurs moyens d'acquisition) en reproduisant les caractéristiques supposées de la vision humaine. Ainsi dotée de sa propre "vision", elle serait alors capable de se mouvoir ou d'interagir avec son environnement ou des objets qu'elle aurait reconnus.

A la différence de la vision humaine, la vision par ordinateur fut longtemps considérée comme étant une méthode ascendante consistant à partir des images perçues uniquement et sans a priori sur la scène ou les objets observés, comme dans le paradigme de Marr, proposé au début des années 1980 par David Marr [141] et qui repose sur trois étapes successives :

1. Traitement bas niveau ou première ébauche : à partir des images d'entrée, construire une description bi-dimensionnelle de celles-ci.
2. Traitement de niveau intermédiaire ou ébauche 2,5D : à partir des descriptions de la première étape, reconstruire la géométrie de la scène à travers des descriptions surfaciques ou volumiques dans un système de coordonnées relatif aux capteurs.
3. Traitement de haut niveau ou représentation de la scène : à partir de l'ébauche 2,5D, construire un modèle intrinsèque (et donc indépendant des capteurs) des objets ou de la scène.

Les problèmes de traitement et d'analyse d'images sont très divers et ne se contentent pas de ceux liés à la vision par ordinateur. Nombre d'entre eux font partie d'une application ou d'un système ne nécessitant pas forcément un niveau d'analyse tel que celui que pourrait offrir la vision artificielle, les rendant plus aisés à mettre en place. Cependant, il est à préciser que certains problèmes de vision par ordinateur peuvent également trouver une application dans un cadre plus spécifique et moins complexe.

Le champ applicatif du traitement et de l'analyse de l'image est des plus vastes. Les problèmes qui en découlent sont donc extrêmement nombreux. Voici néanmoins quelques exemples de problèmes parmi les plus couramment rencontrés :

- la compression d'images, qui est un problème récurrent pour les raisons pratiques de stockage que nécessitent les images dont la résolution ne cesse de croître ;
- la restauration et le débruitage, qui sont nécessaires pour améliorer le rendu d'images détériorées ou dont la qualité d'acquisition a été insuffisante ;
- la reconstruction tomographique, qui est utilisée en imagerie médicale par scanners ou IRM (imagerie par résonance magnétique) ;
- la mesure, la reconnaissance, le suivi ou la recherche de points d'intérêts d'un objet, qui sont requis pour un contrôle qualité ou une vidéo-surveillance automatiques ;
- la stéréoscopie, la reconstruction 3D multi-vues ou le *shape from shading*, qui permettent, à partir d'images, de récupérer des informations de volumes ;
- ...

Ces problèmes sont encore loin d'être tous parfaitement résolus et de nouveaux se posent chaque jour avec la démocratisation croissante des systèmes d'acquisition, de création et de stockage des images depuis l'avènement du "tout numérique" lié aux progrès techniques faits en la matière ces dernières décennies. Ainsi, la classification automatique d'images s'avère devenir un enjeu nécessaire afin de pouvoir se retrouver aisément dans la multitude d'images dont nous disposons.

Il est à préciser que la segmentation d'images, qui consiste à étiqueter différentes régions composants une image, est un problème récurrent faisant souvent partie intégrante d'un traitement plus élaboré. Bien entendu, les critères retenus pour obtenir cet étiquetage sont fortement dépendants du traitement prévu a posteriori. Ceux-ci peuvent très bien être relatifs aux contours d'un objet, à sa différence de couleur ou de texture par rapport à l'arrière-plan, à un a priori sur sa forme ou bien encore à son mouvement.

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 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 [22], leur première apparition est due à Leonhard Euler en 1736. Ce dernier, considéré par beaucoup comme le mathématicien le plus prolifique de tous les temps et un précurseur de la théorie des graphes et de la topologie, fut le premier à apporter une réponse au problème des sept ponts de Königsberg dans [82]. Königsberg (aujourd'hui connue sous le nom Kaliningrad, Fédération de Russie) est une ville traversée par un fleuve, le Pregel, et au milieu duquel se trouvent deux îles reliées entre elles et aux berges par sept ponts (comme illustré dans la figure 2). Le problème était le suivant : peut-on se promener, en revenant à son point de départ, en passant une fois et une seule fois par tous les ponts ? Leonhard Euler prouva à l'aide des prémices de la théorie des graphes sa réponse : non.

Dans un graphe non-orienté, une telle "promenade" peut être représentée par un cycle élémentaire contenant toutes les arêtes du graphe (si ces termes vous sont inconnus, ne paniquez pas, tout cela sera défini et illustré dans le chapitre 1 et plus précisément la section 1.2.2). Un tel cycle est aujourd'hui connu sous le nom de *cycle eulérien*, en référence à son auteur. Un graphe contenant un cycle eulérien est alors appelé graphe eulérien. Leonhard Euler prouva dans [82] qu'un graphe connexe est eulérien si et seulement si le degré de tous ses sommets est paire. Cette condition n'est pas remplie pour le graphe représentant le problème des sept ponts de Königsberg (voir la figure 3).

La résolution de ce problème dans [82] serait donc également la première publication dans le domaine de la recherche opérationnelle (i.e. l'ensemble des méthodes et techniques rationnelles d'analyse et de synthèse de phénomènes d'organisation utilisables pour élaborer de meilleures décisions, notamment en théorie des jeux).

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,...).

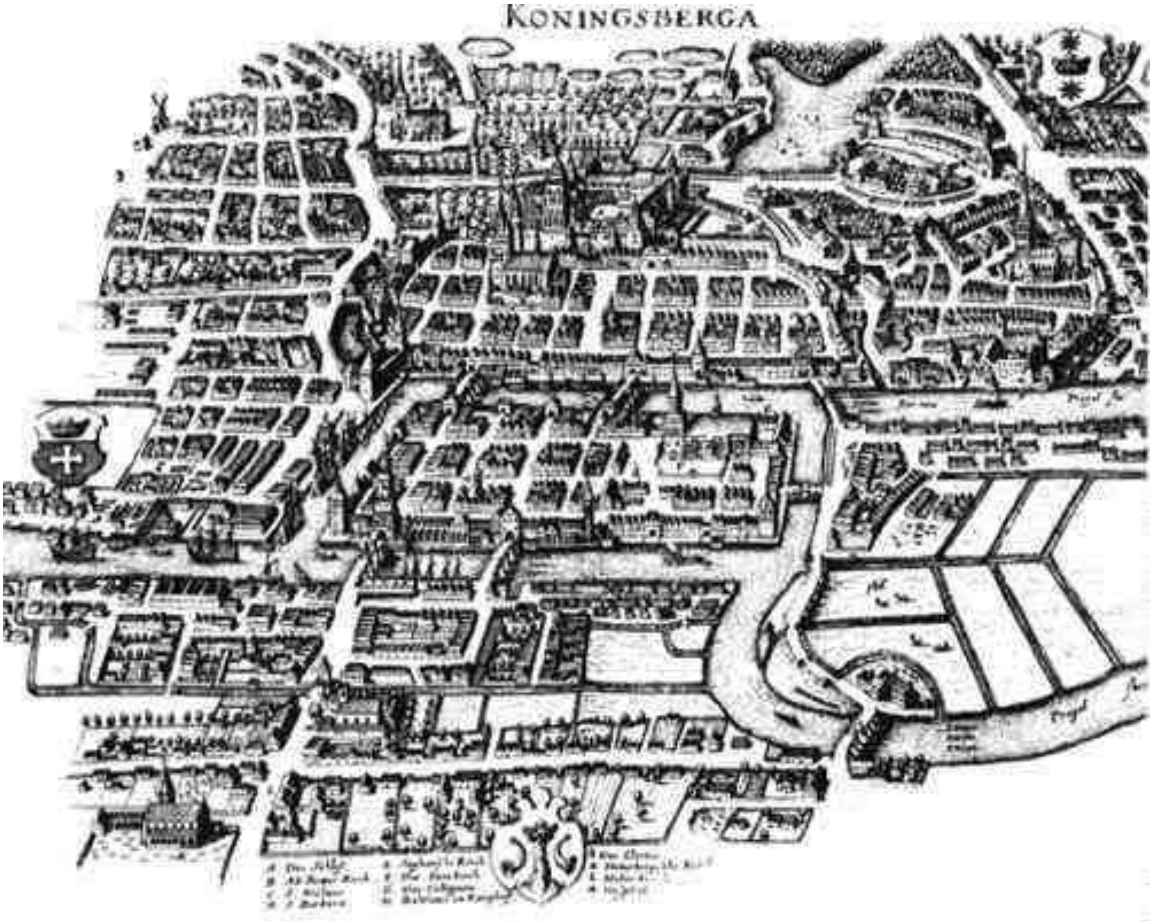
Comme déjà annoncés, les graphes sont également très utilisés en traitement et analyse d'images puisque, comme nous le verrons en détails dans la section 1.5, une image peut être considérée comme un graphe. De nombreux algorithmes sur les graphes ont donc pu être appliqués dans ce domaine et d'autres furent développés spécialement pour. 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.

C'est dans ce cadre que s'inscrit cette thèse et le manuscrit qui en résulte.

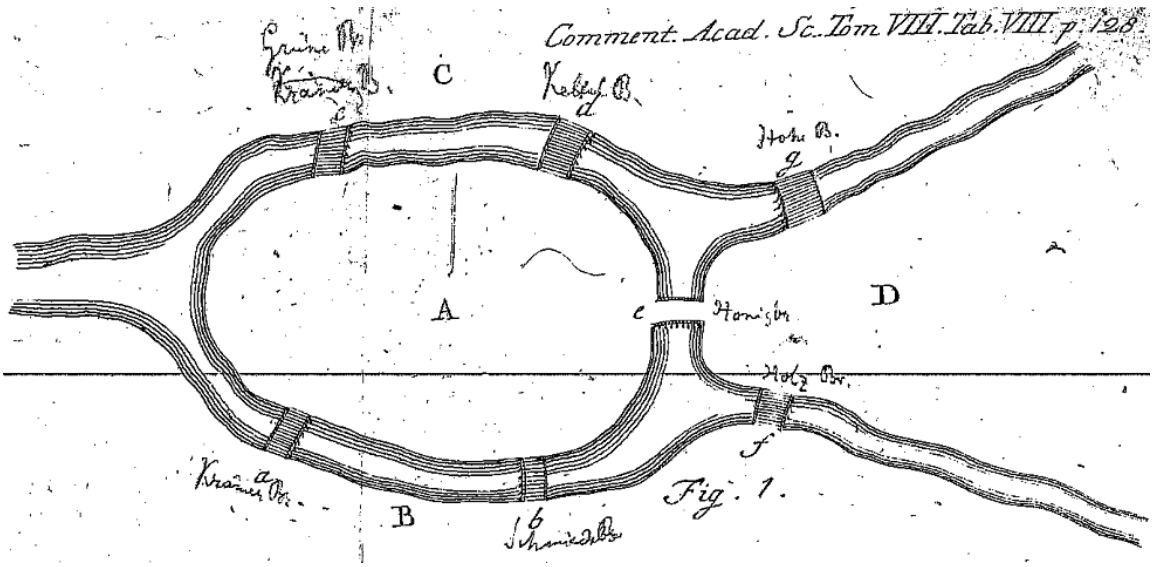
Motivations

Partie théorique

Les paradigmes de segmentation de graphes pondérés sont nombreux et variés. Mais, en premier lieu, il nous a fallu définir ce que nous pouvions considérer comme un résultat possible de segmentation de graphes et en définir les principales propriétés avec leurs avantages et leurs inconvénients. Nous nous sommes ensuite focalisés sur l'étude de certains paradigmes de segmentation répandus offrant l'un ou l'autre de ces résultats. Nous en avons également redéfini d'autres en déduisant de leur définition originelle une nouvelle définition donnant une segmentation au sens où nous l'entendons. Ces paradigmes peuvent être classés en trois catégories :



(a)



(b)

FIGURE 2 – (a) Plan de Königsberg au XVIII^{ème} siècle. (b) Illustration des ponts de Königsberg issue de [82]. Peut-on se promener, en revenant à son point de départ, en passant une fois et une seule fois par tous les ponts ?

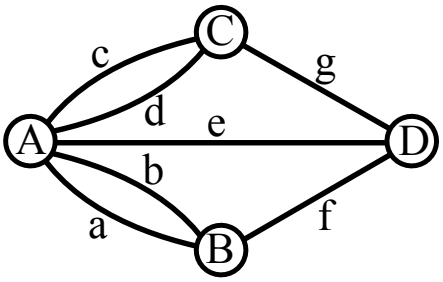


FIGURE 3 – Graphe non-orienté de représentation du problème des sept ponts de Königsberg illustré dans la figure 2 : chaque île ou berge est représenté par un sommet et chaque pont par une arête. Ce graphe n'est pas eulérien. Il est à noter que ce graphe de l'époque ne correspond pas à la définition de graphe non-orienté faite dans la section 1.2.2 puisque deux arêtes différentes ne peuvent relier une même paire de sommets. Cette différence de définition peut cependant être contournée en remplaçant l'arête représentant un pont par un sommet et deux arêtes reliant chacune ce sommet à ceux représentant l'île et la berge que ce pont relie.

- la segmentation par ligne de partage des eaux,
- la segmentation par coupes induites par forêts couvrantes optimales, et
- la segmentation par coupes minimales.

La ligne de partage des eaux (LPE) est un outil de segmentation très utilisé en morphologie mathématique s'inspirant initialement de la ligne de partage des eaux du domaine topographique. De nombreuses définitions différentes co-existent dans la littérature. La plupart offrent en résultat une segmentation mais certaines donnent à la place une application de poids. Il est cependant très aisé de retrouver une segmentation du graphe à partir de celle-ci. Nous avons donc réuni, dans un même cadre de comparaison, certaines des définitions de LPE parmi les plus usités ou les plus récentes et dressé un bilan des propriétés que celles-ci pouvaient partager ou non.

Les forêts couvrantes permettent de partitionner les sommets d'un graphe, offrant ainsi une segmentation de ce dernier. Il existe néanmoins plusieurs méthodes de recherche de forêts couvrantes optimales donnant autant de paradigmes de segmentation différents :

- coupe induite par forêt couvrante de poids minimal,
- coupe induite par forêt couvrante de poids maximal, et
- coupe induite par forêt couvrante de chemins de moindre altitude.

Nous avons comparé ces différents paradigmes afin de mettre en évidence les liens les reliant ou les équivalences existant entre eux ainsi qu'avec une des définitions de LPE préalablement étudiées.

De nombreux algorithmes de recherche de coupe minimale existent mais, à la différence des classes de segmentation précédentes, tous permettent d'avoir un même résultat. Seule l'efficacité liée au temps de calcul mérite alors comparaison dans ce cas. Nous avons cependant cherché à comparer ce type de résultat avec ceux des paradigmes précédemment cités. Ceci nous a ainsi permis de mettre en évidence un lien existant entre coupe minimale et coupe induite par forêt couvrante de poids maximal.

Il est à souligner que, bien souvent par abus de langage, le terme de la littérature anglophone *graph cuts* fait référence aux coupes minimales alors que celles-ci ne sont qu'un des paradigmes existant de coupes de graphes (qui en est la traduction littérale). Le terme anglophone *min-cuts* lui est préférable puisque plus spécifique et donc plus

approprié étant donné les nombreuses possibilités de confusions.

La mise en évidence de ces liens et comparaisons forment la partie théorique du travail réalisé au cours de cette thèse présentée dans la première partie de ce manuscrit.

Ce travail de recherche s'inscrit dans le cadre du projet de l'Agence Nationale de Recherche (ANR) intitulé *SURF* (sous la responsabilité de Hugues Talbot et regroupant le laboratoire CERTIS de l'Ecole des Ponts Paristech, le laboratoire MAS de l'Ecole Centrale de Paris, le laboratoire CEREMADE de l'Université Paris IX Dauphine et le laboratoire A^2SI de l'ESIEE) ayant pour thème général l'étude des notions de surface minimale et de chemin minimal dans les espaces discrets et continus, de leur calcul effectif au moyen d'algorithmes efficaces et de leurs applications en analyse d'images et en vision par ordinateur.

Partie applicative

La seconde partie des travaux effectués durant cette thèse traite de la mise en pratique des coupes minimales dans le cadre de la minimisation d'énergie liée à un champ aléatoire de Markov pour la résolution d'un problème d'étiquetage. Sauf cas particuliers, la résolution d'un tel problème ne peut se faire par une segmentation directe d'un unique graphe mais une approximation du résultat peut être obtenue par coupes minimales successives sur des sous-graphes spécifiques. Deux méthodes ayant des pré-requis de mise en oeuvre différents existent : les $\{\alpha, \beta\}$-*swaps* et les α-*expansions*. Ces méthodes ont été présentées pour la première fois il y moins de dix ans et n'ont depuis cessé d'étendre leur champ de mise en oeuvre et sont devenues des plus populaires.

Le premier problème sur lequel nous nous sommes penchés est celui de la complétion d'image. Il consiste à compléter les régions manquantes, occultées ou endommagées d'une image à partir de ses régions intactes afin d'obtenir en résultat une image qui soit visuellement plausible.

Ce problème, soulevé il y a une décennie, possède diverses méthodes de résolution. Cependant, aucune d'entre elles ne formule la complétion d'image sous la forme d'un problème d'étiquetage à minimiser comme le nôtre qui consiste à attribuer, pour chaque pixel de la zone à reconstruire, la valeur d'un pixel de la zone intacte par copies de petites régions d'images choisies par similitude de voisinage. Néanmoins, devant le très grand nombre d'étiquettes qu'entraîne la résolution de ce problème, un pré-traitement par filtre à particules visant à en réduire le nombre de façon significative fut mis en place.

Notre méthode, à la différence de la plupart des techniques antérieures, permet de compléter avec succès des images fortement texturées.

Le second problème que nous avons traité concerne la création d'un atlas de texture pour la reconstruction 3D de scènes à partir de vues multiples.

La conception, la manipulation et la diffusion d'objets ou de scènes virtuels en 3D se sont fortement répandues ces deux dernières décennies. Ils sont entre autres utiles dans les applications de réalité virtuelle ou augmentée (cette dernière consistant à intégrer des objets virtuels dans des vidéos de scènes réelles), les simulations d'aménagement (d'intérieur, urbain,...), les effets spéciaux dans les films, les jeux vidéo,... Reproduire de façon fidèle des objets ou des scènes de la vie réelle nécessite cependant, la plupart du temps, l'emploi de logiciels de CAO-DAO (Conception Assistée par Ordinateur - Dessin Assisté par Ordinateur). Mais ceux-ci sont souvent réservés à des professionnels ayant eu une formation de modélisation 3D adéquate. De plus, les temps de réalisation sont souvent extrêmement longs si l'on exige un certain degré de réalisme.

C'est pourquoi des méthodes de reconstruction automatiques à partir de différentes

prises de vues de l'objet ou de la scène à reproduire ont commencé à être développées. Une telle méthode nécessite tout d'abord une phase de calibration afin de définir avec précision où les différentes vues ont été prises les unes par rapport aux autres. Celle-ci peut très bien être faite en même temps que l'acquisition des images avec un matériel spécifique ou bien être calculée a posteriori par un algorithme d'autocalibration. Devant les difficultés que nécessite la première méthode, c'est souvent la seconde qui est préférée. Toutefois, les algorithmes d'autocalibration peuvent être source d'erreurs, handicapant ainsi la seconde étape qui consiste en la création du maillage lui-même. Il faut alors arriver à retrouver, dans les différentes vues, les positions d'un même point de l'objet ou de la scène. Une fois ces positions déterminées, et grâce aux données de la calibration des images, il est alors possible de replacer le point dans l'espace 3D et ainsi de construire un maillage reliant l'ensemble des points ainsi positionnés. La dernière étape consiste à restituer les couleurs de l'objet ou de la scène en projetant sur le maillage une texture issue de morceaux des différentes images ayant servi à la reconstruction. C'est dans cette étape que notre méthode intervient.

La constitution de la texture doit être telle que celle-ci ait la meilleure résolution possible afin d'être la mieux détaillée. Cependant, du fait de changements de luminosité ou de paramètres de prises de vues, on observe des différences de teintes pour un même point sur les différentes images. Le choix des images à partir desquelles on crée la texture du maillage doit tenir compte de ces paramètres. C'est pourquoi nous formulons notre problème d'étiquetage de sorte à associer à chaque facette du maillage l'image qui, une fois projetée, offre le plus de détails tout en minimisant les différences de teintes entre deux facettes. Cette minimisation n'est pas toujours suffisante et des frontières disgracieuses peuvent persister sur la texture globale. C'est pourquoi nous procédons à un traitement supplémentaire de mélange de textures par décomposition en bandes de fréquences des images de sorte à créer une transition douce entre les morceaux de textures en provenance de différentes images mais qui ne crée pas pour autant d'artefacts fantômes ou de flou.

Une dernière étape consiste à sauvegarder les morceaux de texture en provenance des différentes vues dans une seule et même image, la plus compacte possible, de sorte à pouvoir être réutilisée aisément avec la majeure partie des programmes de visualisation d'objets 3D.

Cette application s'inscrit dans le cadre du projet de l'Agence Nationale de Recherche (ANR), entrant dans le cadre du Réseau de Recherche et Innovation en Audiovisuel et Multimédia (RIAM) et cofinancé avec Centre National de la Cinématographie (CNC), intitulé *Wired Smart* (sous la responsabilité de Dominique Pouliquen et regroupant le laboratoire CERTIS de l'Ecole des Ponts Paristech, le laboratoire I3S de l'Université de Nice-Sophia Antipolis, le département informatique de l'ENS Ulm et les sociétés Autodesk Realviz et Mikros Image) ayant pour thème général la recherche de solutions matérielles pour l'accélération de problèmes d'analyse d'images en vision par ordinateur.

Organisation et contributions

A l'écriture de ce mémoire, l'objectif était que celui-ci soit pédagogique et autosuffisant, du moins le plus possible. C'est pourquoi nous débutons par des définitions générales sur les graphes, puis, au fur et à mesure des chapitres, nous présentons de nouvelles définitions plus spécifiques à la segmentation de graphes pour finalement nous orienter vers les applications qui se trouvent dans les deux derniers chapitres de ce manuscrit.

La première partie de ce manuscrit présente les résultats théoriques des travaux effectués durant cette thèse.

Les différents paradigmes de segmentation de graphe présentés dans les chapitres 3, 4 et 5 peuvent sembler ne rien avoir en commun d'un prime abord. Nous prouvons cependant que c'est loin d'être le cas puisque des relations d'équivalence ou d'implication existent entre certains d'entre eux dans le cas général ou selon certaines conditions.

Le **chapitre 1**, en premier lieu, dresse la liste des définitions et terminologies de base nécessaire à la manipulation de graphes, orientés et non-orientés. Nous faisons ensuite une aparté généraliste sur les différentes complexité de classifications d'un algorithme avant de retourner dans la domaine des graphes en présentant des définitions et concepts plus complexes sur les graphes qui s'avèrent utiles pour la bonne compréhension de la suite de ce mémoire.

Le **chapitre 2** présente les différentes définitions de segmentation d'un graphe que nous considérons : ensemble frontière de sommets, ensemble frontière d'arêtes et coupe. Une comparaison des différentes propriétés liées à ces trois méthodes est également donnée.

Le **chapitre 3** présente différentes définitions de ligne de partage des eaux parmi les plus usitées ou les plus récentes. Les définitions originelles de LPE dont le résultat est une application de poids plutôt qu'une segmentation sont redéfinies pour permettre la comparaison de leurs propriétés dans le cadre de la segmentation de graphes. Ceci nous permet de mettre en évidence des liens ou équivalences existant entre ces différentes définitions qui constituent les nouveautés de ce chapitre.

Le **chapitre 4** présente les définitions de forêts couvrantes de poids extremum et de chemins de moindre altitude. Les propriétés unissant ces deux définitions sont rappelées avant de présenter les liens entre les coupes induites par chacune de ces définitions et la LPE d'arêtes relative qui constituent les nouveautés de ce chapitre.

Le **chapitre 5** présente différents algorithmes 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 avant d'en présenter une généralisation, appelée *multiway cut* minimale, pour des coupes segmentant un graphe non-orienté en plus de deux régions mais dont seul un résultat approché peut être trouvé. Enfin, nous présentons le lien entre *multiway cut* minimale et coupe induite par forêt couvrante de poids maximal qui constitue la nouveauté de ce chapitre.

La seconde partie présente la mise en application des coupes minimales dans le cadre de la minimisation d'énergie liée à un champ aléatoire de Markov pour la résolution d'un problème d'étiquetage ainsi que les deux applications en découlant mises en oeuvre durant cette thèse.

Le **chapitre 6** 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 les différentes techniques de résolution reposant sur les coupes minimales de graphes en commençant par le cas simple et optimal d'un étiquetage binaire puis en montrant comment celui-ci a pu être étendu sous la forme de deux méthodes, l'$\{\alpha, \beta\}$-*swap* et l'α-*expansion*, pour le cas général d'un nombre quelconque d'étiquettes mais avec un résultat approché. Il est à préciser que les constructions de graphes proposées ici varient des contructions originelles en traitant le cas où chaque site ne peut pas recevoir toutes les étiquettes du problème et permettant également de réduire le nombre de sommets et d'arcs du graphe sur lequel la coupe minimale doit être exécutée, accélérant ainsi sa détermination.

Le **chapitre 7** présente notre application de complétion d'image, la *renaissance d'image*.

Nous définissons tout d'abord ce problème sous la forme d'un problème d'étiquetage associé à une énergie à minimiser. Nous présentons ensuite comment, à l'aide d'un filtre à particules, nous réduisons considérablement le nombre d'étiquettes possible pour chaque site avant de procéder à sa résolution par la méthode de minimisation par *α-expansions*. Enfin, nous présentons les résultats obtenus avec notre méthode pour la complétion d'image texturée ainsi que sur des exemples classiques afin de permettre la comparaison.

Le **chapitre 8** présente notre application de création d'un atlas de texture pour la reconstruction 3D de scènes à partir de vues multiples. Nous définissons tout d'abord ce problème sous la forme d'un problème d'étiquetage associé à une énergie à minimiser. Ensuite, afin d'améliorer le rendu du résultat sur lequel des frontières entre fragments de textures peuvent demeurer visibles, nous présentons une méthode de mélange d'images basée sur la décomposition en bandes de fréquences, par la construction de pyramides d'images gaussiennes et laplaciennes, que nous avons généralisée afin de la rendre applicable pour un mélange de plus de deux images tout en s'affranchissant des contraintes de tailles originelles. Enfin, nous appliquons ensuite cette technique à la texture obtenue par minimisation avant d'expliquer comment créer l'atlas résultant. Enfin, nous présentons les résultats obtenus avec notre méthode pour la texture de deux larges scènes extérieures reconstruites : l'Aiguille du Midi (Mont-Blanc, Chamonix, France) et le château d'Ettlingen (Ettlingen, Allemagne).

Première partie

Graphes et théories

Généralités sur les graphes

1

Ce premier 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 procédons alors, en guise d'aparté, à un court rappel sur la complexité des algorithmes et leur classification qui seront utilisées pour les comparer dans la suite de ce mémoire. 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.

Sommaire du chapitre

Figures du chapitre

1.1 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.
  $A \cup B = \{x \mid x \in A \text{ ou } x \in B\}$
- L'**intersection** des ensembles A et B, noté $A \cap B$, est l'ensemble des éléments qui sont dans A et dans B.
  $A \cap B = \{x \mid x \in A \text{ et } x \in B\}$
- 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.
  $A \setminus B = \{x \mid x \in A \text{ et } x \notin B\}$
- 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 $\overline{A}^B$ ou plus simplement $\overline{A}$ 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, $A \times B \neq B \times A$.
  $A \times B = \{(a, b) \mid a \in A \text{ et } b \in B\}$

Exemple 1.1.
Soient les ensembles A et B tels que :

- $A = \{a, b, d\}$,
- $B = \{b, c\}$.

Nous avons alors :

- $|A| = 3$,

- $|B| = 2$,
- $\{b, d\} \subset A$,
- $\{b, d\} \not\subset B$,
- $A \cup B = \{a, b, c, d\}$,
- $A \cap B = \{b\}$,
- $A \setminus B = \{a, d\}$,
- $\mathcal{P}(A) = \{\emptyset, \{a\}, \{b\}, \{d\}, \{a, b\}, \{a, d\}, \{b, d\}, \{a, b, d\}\}$,
- $\mathcal{P}(B) = \{\emptyset, \{b\}, \{c\}, \{b, c\}\}$,
- $\overline{b}^A = \{a, d\}$,
- $\overline{b}^B = \{c\}$,
- $A \times B = \{(a, b), (a, c), (b, b), (b, c), (d, b), (d, c)\}$.

Dans ce mémoire, nous noterons $[k; \ell]$ le sous-ensemble d'entiers tel que $[k; \ell] = \{i \in \mathbb{Z}; k \leq i \leq \ell\}$.

1.2 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.

1.2.1 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 1.2.
Un exemple de graphe orienté se trouve dans la figure 1.1 (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.*

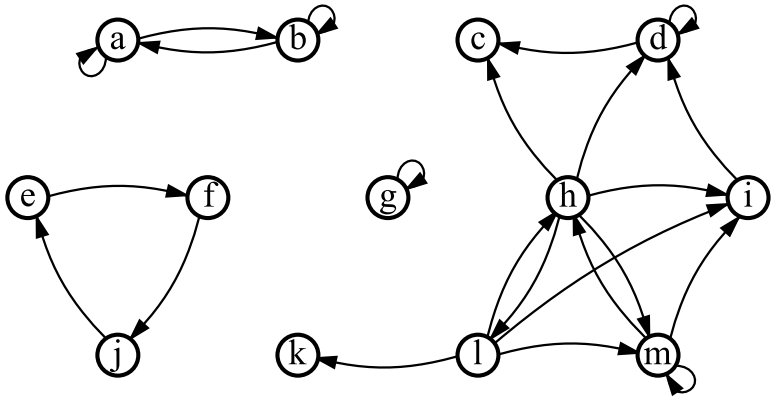


FIGURE 1.1 – Graphe orienté.

1.2.1.1 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 1.2.

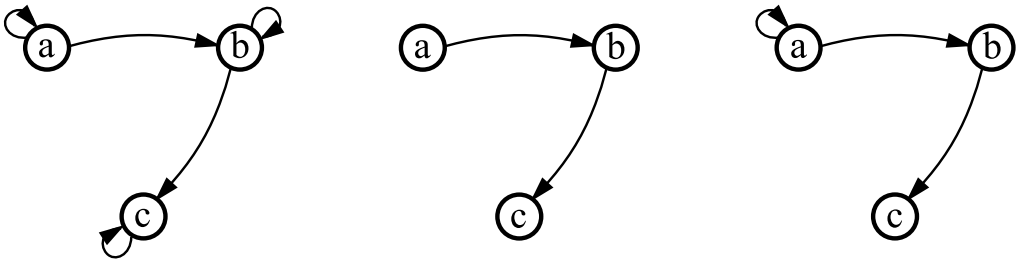
1.2.1.2 Graphe symétrique et fermeture symétrique

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 a été inversée par rapport au graphe d'origine.

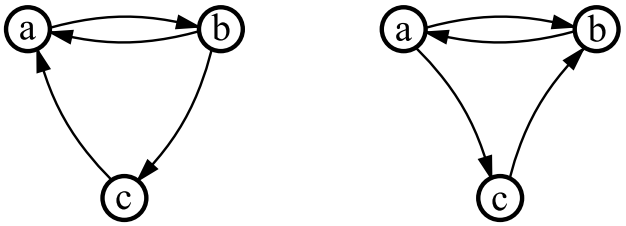
Soit $a \in A$. On note a^{-1} l'arc de A^{-1} symétrique de a.

Un exemple de cette définition se trouve dans la figure 1.3.



(a) (b) (c)

FIGURE 1.2 – (a) Graphe orienté réflexif ; (b) Graphe orienté antiréflexif ; (c) Graphe orienté non-réflexif et non-antiréflexif.



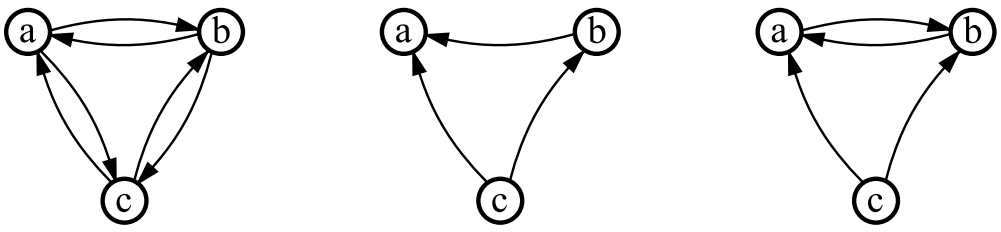
(a) (b)

FIGURE 1.3 – (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 1.4.



(a) (b) (c)

FIGURE 1.4 – (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 1.1 se trouve dans la figure 1.5.

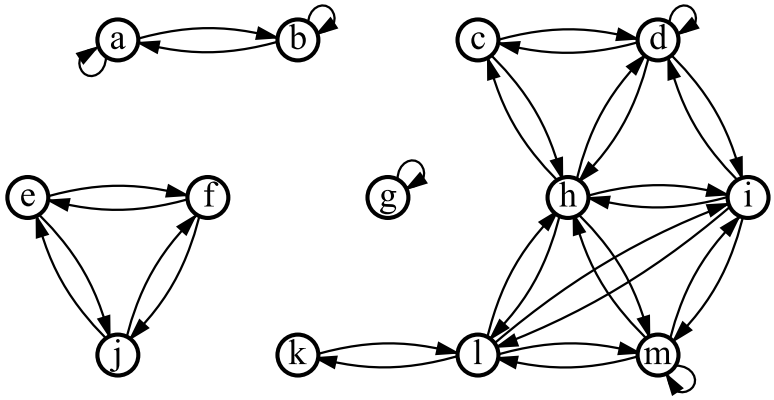


FIGURE 1.5 – Graphe orienté symétrique qui est la fermeture symétrique du graphe de la figure 1.1.

1.2.2 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 1.3.
Un exemple de graphe non-orienté se trouve dans la figure 1.6 (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.*

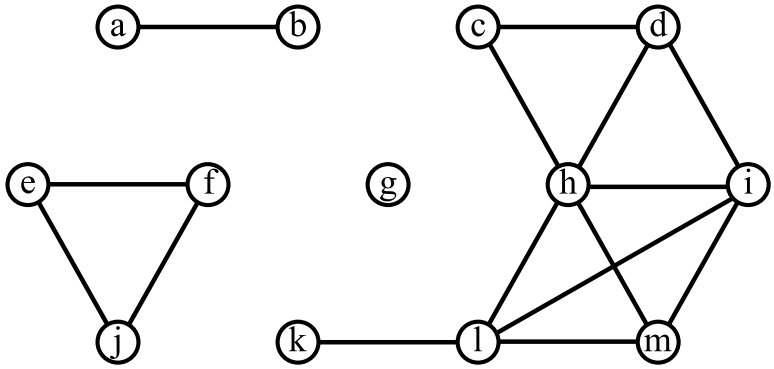


FIGURE 1.6 – Graphe non-orienté.

1.2.3 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 anti-réflexif 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 1.1 et sa fermeture symétrique, représentée sur la figure 1.5, qui sont tous deux associés au graphe non-orienté de la figure 1.6.

1.3 Complexité d'un algorithme et classes de problèmes

Avant d'aller plus loin et d'aborder la présentation des graphes à proprement parler, il s'avère utile de faire un bref aparté pour présenter la notion de complexité d'un algorithme tel que ceux qui seront présentés dans la suite de ce mémoire. En effet, parmi les définitions répertoriées dans ce document, un certain nombre est accompagné d'un ou plusieurs algorithmes permettant d'obtenir pour résultat l'objet de la définition en question à partir d'un graphe donné. Un algorithme est l'énoncé d'un moyen de résolution par calculs d'un problème donné grâce à une suite d'opérations. Mais pour juger si un algorithme est meilleur qu'un autre, il fallut mettre en place un moyen de comparer leur efficacité. Le critère principal qui fut choisi est celui de la rapidité d'exécution. Toutefois, une évaluation de celle-ci nécessite une implémentation sous forme de programme dont on chronomètre une exécution sur un ordinateur avec un jeu de données suffisamment grand pour obtenir un temps significatif. Cette méthode est donc fortement dépendante du langage de programmation, des données et enfin du matériel utilisés, rendant ainsi la comparaison peu aisée puisque, pour être objective, les conditions exactes d'exécutions doivent être reproduites.

Cette section vise à présenter rapidement ce que l'on entend par complexité d'un algorithme ainsi que les notations qui la représentent.

1.3.1 Principe et notation de la complexité d'un algorithme

Dans la série de livres *The art of computer programming*, dont la parution débuta en 1968, Donald Knuth fut un des premiers à faire cette évaluation en ayant recours à une approche indépendante des facteurs matériels relative en utilisant comme unité de comparaison le nombre d'opérations élémentaires effectuées en fonction de la taille des données d'entrée au lieu d'unité de mesure temporelle.

On appelle **opération élémentaire** une opération simple effectuée en temps constant, telle que, par exemple :

- un accès à une cellule mémoire,
- une comparaison de valeurs,
- une opération arithmétique (sur des valeurs à codage de taille fixe),
- une opération sur des pointeurs,
- ...

La **taille des données** sur lesquelles s'applique l'algorithme est représentée par un ou plusieurs entiers liés aux nombres d'éléments des données d'entrée. Cela peut être, par exemple :

- le nombre d'éléments dans un tableau,
- le nombre de sommets et le nombre d'arcs/arêtes d'un graphe,
- ...

La **complexité** d'un algorithme consiste ainsi à définir une fonction donnant le nombre d'opérations élémentaires en fonction des entiers représentant la taille des données d'entrée. Néanmoins, ce nombre d'opérations peut varier substantiellement pour des données de même taille. L'analyse de la complexité peut ainsi se faire soit pour le plus mauvais cas (celui qui offre le temps d'exécution le plus long), soit pour la moyenne des cas (si le plus mauvais cas ne se produit que rarement par exemple). Le plus souvent, c'est la complexité du pire cas qui est présentée avec un algorithme.

L'efficacité d'un algorithme n'est souvent considérée que pour des données d'entrée de très grande taille afin d'avoir une idée de l'**ordre de grandeur** d'un temps d'exécution. Ainsi, l'analyse de la complexité d'un algorithme se fait de manière asymptotique. On utilise pour cela les **notations de Landau**, introduites pour la première fois par Paul Bachmann en 1894 dans son second volume sur la théorie des nombres [18] mais développées par Edmund Landau en 1909 dans son livre sur la distribution des nombres premiers [132] duquel elles tirent leur nom. Fondamentalement, cela consiste à étudier la croissance de la fonction représentant le nombre d'opérations élémentaires pour de grandes tailles de données.

Soient n la taille des données d'entrée et $f(n)$ le nombre d'opérations élémentaires en fonction de n. Si l'on considère, par exemple, $f(n) = 4n^3 + 10n^2 + 15n + 120$, alors f à une croissance en n^3. On dira alors que la complexité de l'algorithme correspondant est en n^3 et on notera cette complexité $O(n^3)$. On utilise la lettre *"O"* pour symboliser la complexité car la croissance d'une fonction est aussi appelée *ordre*.

De manière générale, une complexité est considérée comme *meilleure* qu'une autre si sa croissance est plus faible. Ainsi, voici quelques exemples courants de complexités avec leurs notations, classées de la *meilleure* à la *moins performante* :

- complexité constante (indépendante de la taille des données), notée $O(1)$;
- complexité logarithmique, notée $O(\log(n))$;
- complexité polylogarithmique, notée $O((\log(n)^p)$;
- complexité linéaire, notée $O(n)$;
- complexité quasi-linéaire ou linéarithmique, notée $O(n\log(n))$
- complexité quadratique, notée $O(n^2)$;

- complexité cubique, notée $O(n^3)$;
- complexité polynomiale ou géométrique, notée $O(n^p)$;
- complexité quasi-polynomiale, notée $O(n^{\log(n)})$;
- complexité exponentielle, notée $O(2^n)$;
- complexité factorielle, notée $O(n!)$.

Bien entendu, dans le cas où les données d'entrée sont représentées par plusieurs entiers, le principe reste le même, faisant simplement intervenir plusieurs paramètres au lieu d'un seul. On pourra alors avoir, par exemple, une complexité linéaire en $O(n+m)$.

Le chapitre 2 de [53] donne davantage de détails sur ce thème.

Il est cependant à noter que, en pratique, un algorithme avec une complexité considérée moins bonne peut très bien obtenir des résultats plus rapidement pour certaines tailles de données. En effet, jusqu'à quel point une exécution de $1000n^2$ opérations élémentaires, ayant donc une complexité en $O(n^2)$, est-elle réellement meilleure qu'une exécution de $5n^3$ opérations élémentaires, ayant donc une complexité en $O(n^3)$?

Une autre limitation de cette représentation de la complexité est qu'elle ne prend pas en compte la quantité d'espace mémoire utilisée pour effectuer les calculs qui est également un critère déterminant dans le choix d'un algorithme. Entre deux algorithmes de complexité identiques on choisira bien évidemment celui occupant le moins de place. On pourra également être amené à choisir un algorithme requérant moins de place mémoire et ayant une complexité moins bonne si l'espace mémoire allouant est un facteur limitant.

Un autre point à prendre en compte est la spécificité de l'algorithme par rapport aux données. En effet, un algorithme à la complexité adapté aux données du problème peut très bien être exécuté en un nombre d'opérations élémentaires plus faible qu'un algorithme non-spécialisé ayant une complexité considérée comme meilleure. En effet, dans le cas de l'algorithme spécialisé, la complexité étant déterminée par son pire cas, si celui ne se produit que rarement en pratique, il sera alors plus avantageux de l'utiliser. Nous verrons un tel cas dans le chapitre 5.

1.3.2 Différentes classes de complexités

L'analyse de la complexité des algorithmes et leur classification sont étroitement liées à des modèles de calculs. L'un de ceux les plus utilisés est celui des machines abstraites, du type du modèle proposé par Alan Turing en 1937 [199, 200]. Dans une telle machine, un algorithme est composé d'étapes élémentaires. A chaque étape, pour un état donné de la mémoire de la machine, une action élémentaire est choisie parmi celles possibles. On distingue alors les deux types de machines suivantes :

- les **machines déterministes**, qui sont les machines où l'action à effectuer est déterminée de façon unique par l'état courant de celle-ci, ainsi chaque action possible est unique ;
- les **machines non-déterministes**, qui sont les machines où il peut y avoir plusieurs choix d'actions possibles à effectuer.

Nous présentons ci-après les classes de complexité qui pourront être rencontrées dans ce mémoire, classées par ordre de complexité croissante.

1.3.2.1 Classe P

On appelle **classe P** (pour **P**olynomiale) l'ensemble des problèmes pour lesquels une solution peut être obtenue sur une machine déterministe avec un algorithme polynomial.

Exemple 1.4.
La recherche du plus grand élément d'un ensemble d'entiers est un problème appartenant à la classe P.

Généralement, on admet que les problèmes qui sont dans P sont facilement solubles.

1.3.2.2 Classe NP

On appelle **classe NP** (pour **N**on-déterministe **P**olynomiale) l'ensemble des problèmes pour lesquels une solution peut être obtenue sur une machine non-déterministe avec un algorithme polynomial.

De façon équivalente, on dit que c'est l'ensemble des problèmes pour lesquels il existe un algorithme polynomial sur une machine déterministe capable de tester la validité d'une solution (ce test appartient donc à la classe P). Ce sont donc les problèmes qui peuvent être résolus en énumérant l'ensemble des solutions possibles et en les vérifiant une à une (à l'aide de l'algorithme polynomial de test).

Exemple 1.5.
La recherche, parmi un ensemble d'entiers, d'un sous-ensemble dont la somme est nulle est un problème appartenant à la classe NP. En effet, tester si une somme d'entiers est nulle est un problème de la classe P. Par contre, sélectionner un sous-ensembe d'entiers (pour lui faire passer ce test) est non-déterministe puisqu'il faut les "choisir".

Il est à noter que, par définition nous avons $P \subseteq NP$. Toutefois, bien que non prouvé, il est fortement supposé par la communauté scientifique que $P \neq NP$ (comme expliqué plus en détails dans la sous-section 1.3.2.6). Cette question, posée en 1970 par Stephen Cook et Leonid Levin en 1970 reste donc ouverte.

1.3.2.3 Classe APX

On appelle **classe APX** (pour l'abréviation d'**approximable**) l'ensemble des problèmes de la classe NP ayant un algorithme d'approximation bornée par une constante et s'exécutant en temps polynomial.

Un algorithme d'approximation est un algorithme de k**-approximation** si la solution donnée par l'algorithme vaut au plus k fois celle de la solution optimale. La constante k s'appelle alors le **ratio d'approximation**. Selon que le problème est une minimisation ou une maximisation, le résultat de l'approximation est, respectivement, k fois plus grand ou k fois plus petit que le résultat optimal.

Un algorithme d'approximation a recours à des choix stratégiques lui permettant de trouver une solution convenable la plus proche possible d'une solution optimale (et qui peut s'avérer exacte dans certains cas) dans un temps raisonnable sans avoir à tester toutes les combinaisons possibles. Ces choix sont très dépendants du problème traité et constituent ce qu'on appelle une **heuristique**.

On appelle algorithme **glouton** un algorithme qui, à chaque étape de son exécution, fait un choix local optimal dans l'espoir d'obtenir un résultat global optimal. Dans les cas où l'algorithme ne fournit pas systématiquement la solution optimale, on a alors affaire à une heuristique gloutonne. Ce type d'heuristique est fréquemment utilisé puisqu'un algorithme glouton ne revient jamais sur un choix antérieur qu'il a fait et qu'il est donc souvent rapide d'obtenir une solution même si celle-ci n'est pas optimale.

1.3.2.4 Classe MAX SNP

Si, pour tout k, un problème possède un algorithme de k-approximation s'exécutant en temps polynomial, alors on dit que ce problème possède un **schéma d'approximation en temps polynomial** (**PTAS** pour ***P**olynomial **T**ime **A**pproximation **S**cheme* en anglais).

On appelle **classe MAX SNP** l'ensemble des problèmes de la classe APX n'ayant pas de PTAS.

Cette classe de complexité fut introduite en 1988 par Christos Papadimitriou et Mihalis Yannakakis dans [161].

1.3.2.5 Problèmes difficiles

Soit C une classe de complexité (parmi celles exposées ci-avant par exemple).

De façon informelle, un problème est dit **C-difficile** s'il est au moins aussi difficile à résoudre que tout problème de la classe C.

Formellement, on définit la réduction d'un problème Π_1 à un problème Π_2 comme étant un algorithme (de complexité inférieure à celle de la classe C) transformant toute instance (aussi appelée jeu de données) du problème Π_2 en une instance du problème Π_1. Ainsi, si on sait résoudre Π_1 avec un certain algorithme, on sait également résoudre Π_2 avec le même algorithme. On dit alors qu'un problème Π est C-difficile si pour tout problème Π' de C, Π' se réduit à Π.

Exemple 1.6.
Si un problème Π est NP-difficile, cela signifie qu'il y a un algorithme dans P permettant de réduire tout problème Π' de classe NP à Π.

1.3.2.6 Problèmes complets

Soit C une classe de complexité (parmi celles exposées ci-avant par exemple).

Un problème est dit **C-complet** si :

- il est dans C, et
- il est C-difficile.

Les problèmes C-complets sont donc les problèmes de la classe C les plus difficiles à résoudre.

Ainsi, les problèmes NP-complets ne possèdent pas d'algorithme polynomial capable de trouver une solution. Il est alors souvent préférable de chercher un algorithme obtenant une approximation du résultat avec une meilleure complexité (si possible polynomiale). En 1972, Richard Manning Karp présenta dans [115] 21 problèmes types prouvés comme étant NP-complets.

Exemple 1.7.
Le problème de recherche d'un sous-ensemble d'entiers de somme nulle (voir exemple 1.5) est un problème NP-complet.
Le problème du voyageur de commerce, qui consiste à minimiser le trajet d'un voyageur de commerce devant passer par n villes avant de retourner à son point de départ (et qui est donc facilement représentable sur un graphe comme nous le verrons avec les définitions de ce chapitre), est NP-complet également. Le nombre de chemins possibles passant par 69 villes différentes est déjà un nombre à 100 chiffres (pour comparaison, le nombre d'atomes dans tout l'univers connu peut être représenté par un nombre à "seulement" 80 chiffres...), d'où la nécessité d'un algorithme d'approximation.

Le fait que les problèmes NP-complets ne soient pas solubles en temps polynomial est la raison qui laisse à penser que $P \neq NP$. En effet, par définition si un seul problème NP-complet pouvait être résolu en temps polynomial, alors, par réduction, tous les problèmes

NP-complets le pourraient. Pourtant, malgré des années de recherche, aucun algorithme polynomial n'a été trouvé pour un quelconque problème classé NP-complet.

1.4 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 $G = (S(G), A(G))$.

1.4.1 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.

1.4.2 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 1.7.

1.4.3 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.

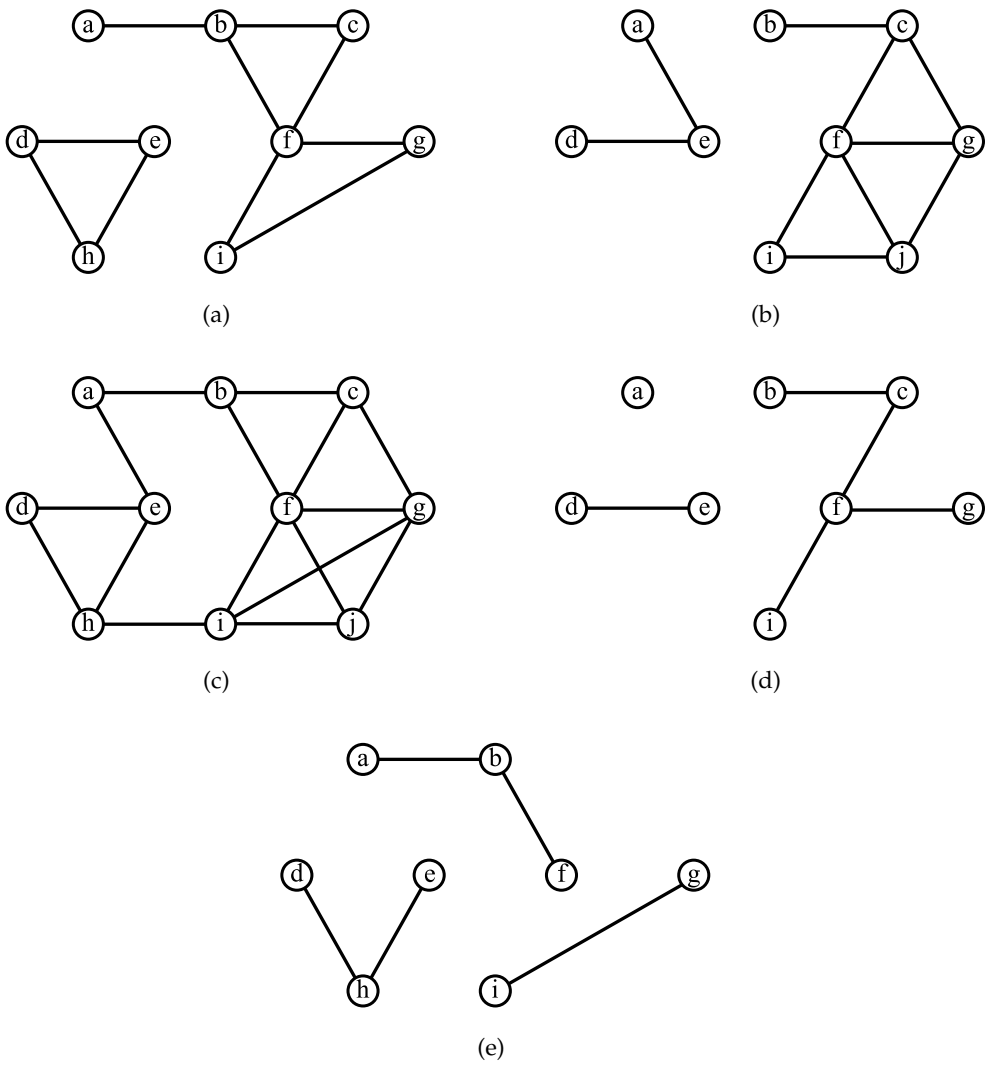


FIGURE 1.7 – (a) Graphe non-orienté G_1 ; (b) Graphe non-orienté G_2 ; (c) $G_1 \cup G_2$; (d) $G_1 \cap G_2$; (e) $G_1 \setminus G_2$.

Remarque 1.8. *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.

Exemple 1.9.
Les composantes connexes des graphes représentés dans les figures 1.1, 1.5 et 1.6 sont les sous-graphes induits par les ensembles de sommets suivants :
- $\{a, b\}$,
- $\{c, d, h, i, k, l, m\}$,
- $\{e, f, j\}$,
- $\{g\}$.

1.4.4 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 1.10.
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 $G_\emptyset$.

1.4.5 Adjacence et fonction successeur

Dans un graphe G, la relation d'adjacence sur les sommets peut être représentée par une application Γ 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).

1.4.6 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 :

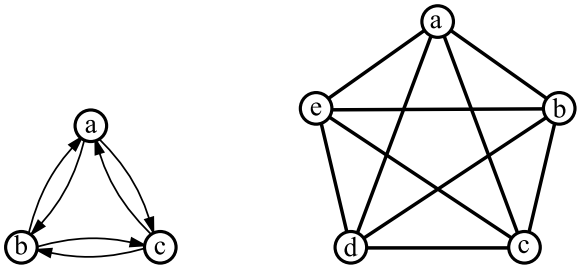
- dans le cas des graphes orientés, $\forall x, y \in S', (x, y) \in A$ et $(y, x) \in A$;
- dans le cas des graphes non-orientés, $\forall x, y \in S', \{x, y\} \in A$.

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

Exemple 1.11.
Dans la figure 1.6, 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 1.8.



(a) (b)

FIGURE 1.8 – (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 [115] (où il présentait 21 problèmes NP-complets).

1.4.7 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 G_2.

Remarque 1.12.
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.

Exemple 1.13.
$(\{1,2\}, \{(1,2),(2,1)\}) \sim (\{a,b\}, \{(a,b),(b,a)\})$

1.4.8 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 1.14 ci-dessous, mais elle fut ensuite étendue aux graphes orientés (voir définition 1.20). Même si cette dernière définition peut également servir pour le cas des graphes non-orientés, la définition 1.14 est plus aisée à comprendre pour l'introduction de ce concept.

1.4.8.1 Graphe dérivé non-orienté

Définition 1.14 (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.

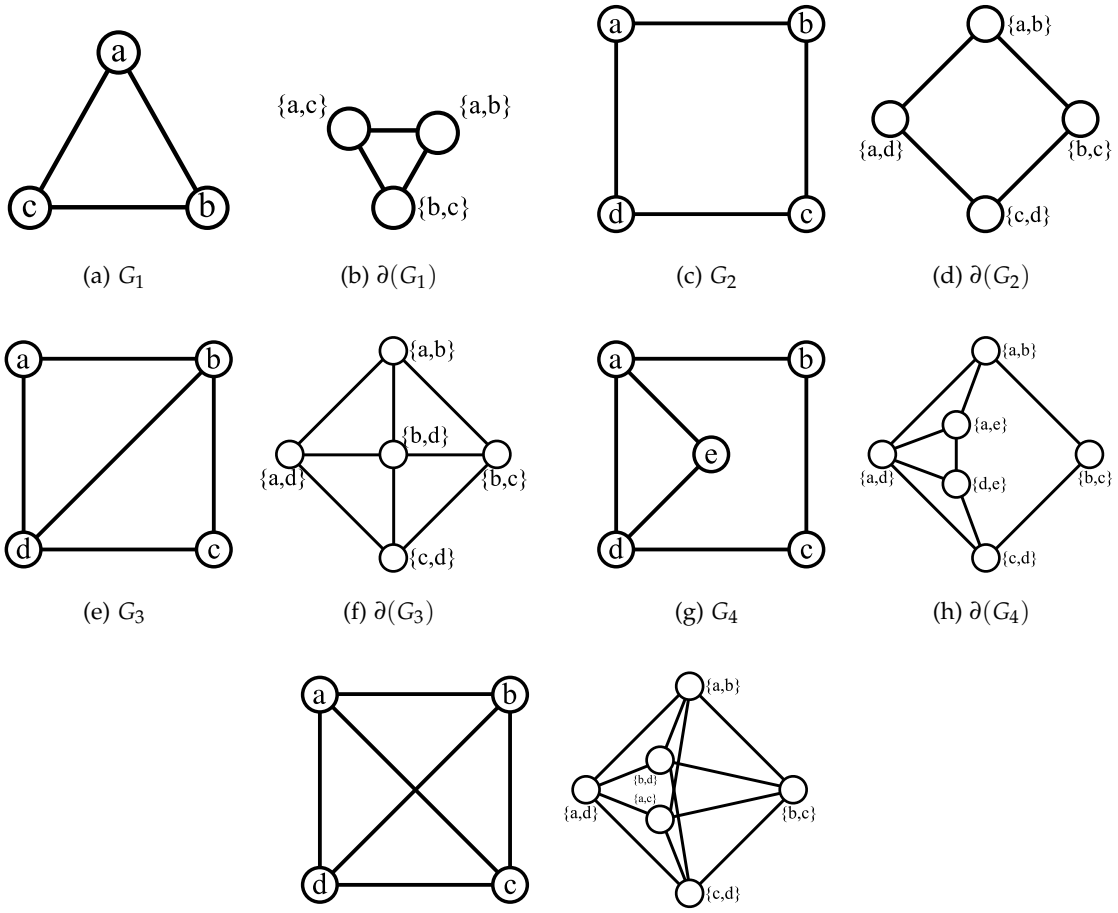
Des exemples de graphes dérivés non-orientés se trouvent dans la figure 1.9.

Remarque 1.15.
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. Dans [209], Hassler Whitney prouva qu'il ne peut y avoir qu'une seule partition de ce type possible (au graphe "triangulaire" près - voir la remarque 1.17).
Ceci se retrouve à travers la propriété 1.16.

Propriété 1.16 ([175], chapitre 17, propriété 2)**.**
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 1.17.
Hassler Whitney montra qu'un graphe "triangulaire" (figures 1.9a et 1.9b) 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



(a) G_1 (b) $\partial(G_1)$ (c) G_2 (d) $\partial(G_2)$

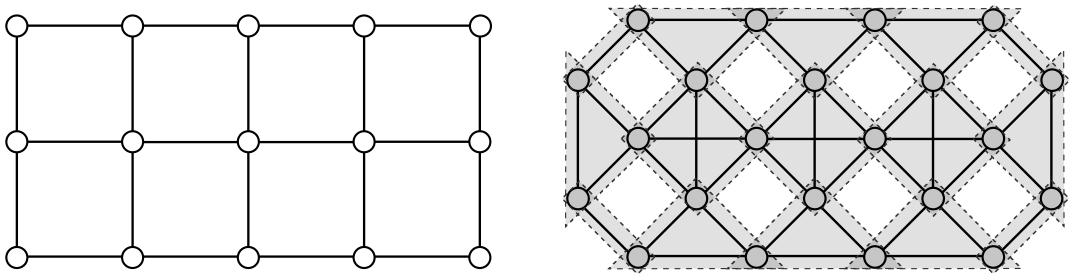
(e) G_3 (f) $\partial(G_3)$ (g) G_4 (h) $\partial(G_4)$

(i) G_5 (j) $\partial(G_5)$

FIGURE 1.9 – Des graphes non-orientés (a, c, e, g, i) et leurs graphes dérivés non-orientés respectifs (b, d, f, h, j).

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 1.10), à l'exception toujours du graphe "triangulaire".



(a) (b)

FIGURE 1.10 – (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 [179], et Philippe G. H. Lehot, en 1974 [133], 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 [69], 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 au nombres de sommets qui lui sont adjacents.

Une caractérisation alternative des graphes dérivés fut prouvée par Lowell W. Beineke en 1968 [21]. 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 1.18).

Théorème 1.18 ([21]).
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 1.11.

Remarque 1.19.
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 1.18 (voir figure 1.12).

1.4.8.2 Graphe dérivé orienté

Définition 1.20 (Graphe dérivé orienté)**.**
*Le graphe $G' = (S', A')$ est le graphe **dérivé** du graphe $G = (S, A)$ si :*

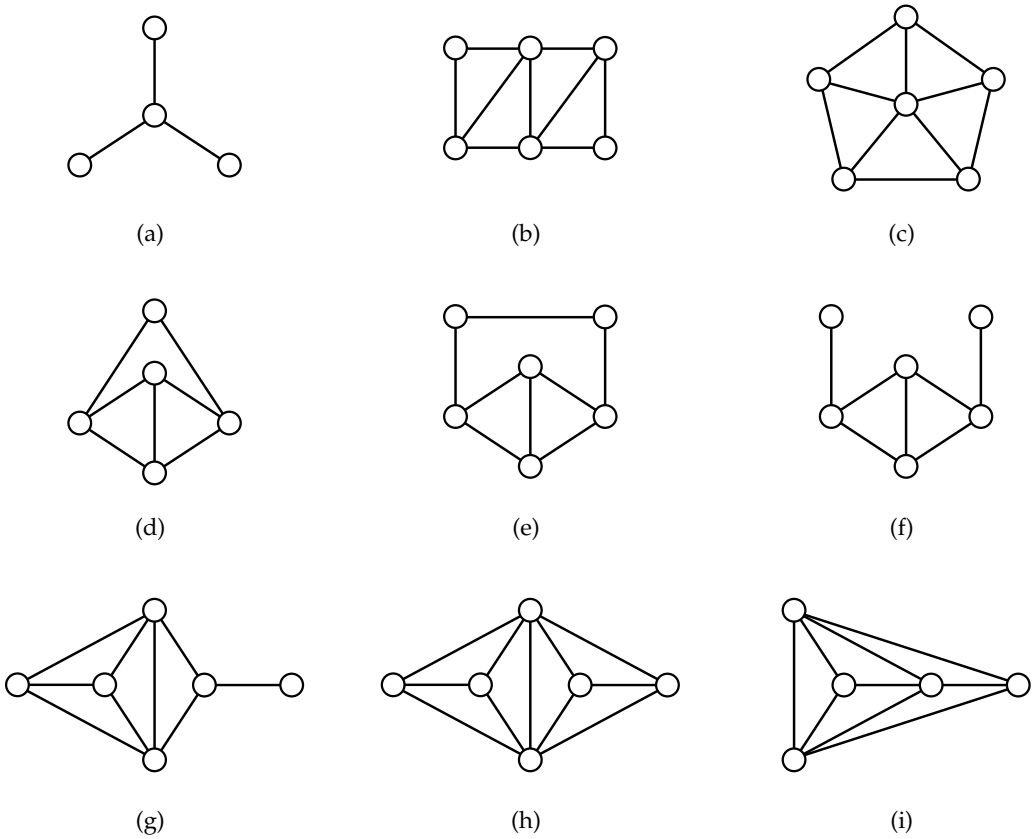


FIGURE 1.11 – Les neuf graphes minimaux non-dérivés.

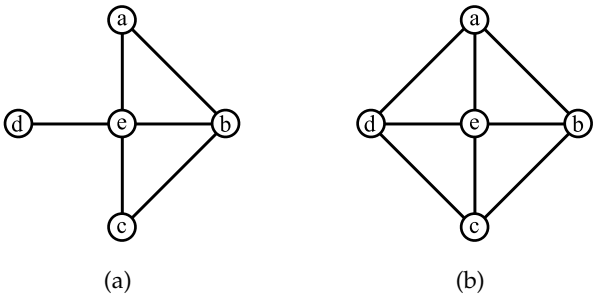


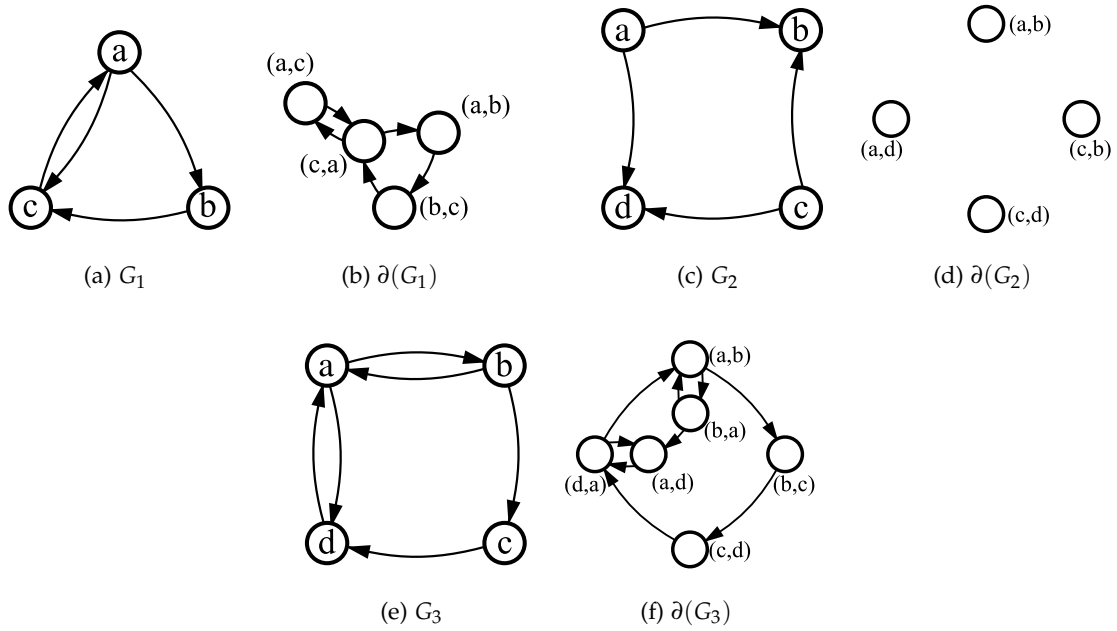
FIGURE 1.12 – (a) Graphe non-dérivé car ayant pour sous-graphe induit par les sommets $\{a, c, d, e\}$ un des graphes non-dérivés minimaux (Figure 1.11a) ; (b) Graphe dérivé (voir figuresé 1.9e et 1.9f) malgré qu'il contienne en sous-graphe un des graphes non-dérivés minimaux (figure 1.11a).

– *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.

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 1.13.



(a) G_1 (b) $\partial(G_1)$ (c) G_2 (d) $\partial(G_2)$

(e) G_3 (f) $\partial(G_3)$

FIGURE 1.13 – Des graphes orientés (a, c, e) et leurs graphes dérivés orientés respectifs (b, d, f).

1.4.8.3 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 1.21.
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 1.22.
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 1.23.
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 1.23.
La remarque 1.21 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 1.23, 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 $X' = (\partial(A(X)),\partial(S(X)))$. Bien sûr, si $X = (s,\emptyset)$ alors $\partial(X) = G_\emptyset$.

Propriété 1.24.
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. $|\mathcal{C}(\partial(X))| = |\mathcal{C}(X)|$).

Preuve de la propriété 1.24.
D'après la propriété 1.23, nous savons que la dérivée de chaque composante connexe de X est connexe. Il reste donc à vérifier que ces composantes connexes dérivées ne sont pas liées dans $\partial(X)$.
Soient X_1 et X_2 deux composantes connexes de X. Supposons que $\partial(X_1)$ et $\partial(X_2)$ sont liées dans $\partial(X)$. Il y a alors deux possibilités :
- soit $\partial(X_1) \cap \partial(X_2) \neq G_\emptyset$,
- soit il existe une ou plusieurs autres composantes connexes X'_i (avec $i \in [1; j]$) de X, de telle sorte que leurs dérivées $\partial(X'_i)$ lient $\partial(X_1)$ et $\partial(X_2)$, auquel cas nous aurions $\partial(X_1) \cap \partial(X'_1) \neq G_\emptyset$ et $\partial(X'_j) \cap \partial(X_2) \neq G_\emptyset$.

Il suffit donc de prouver que les dérivées de deux composantes connexes quelconques distinctes de X ont une intersection vide, autrement dit que l'intersection de leurs ensembles de sommets est vide. Or par, définition, les dérivées de deux sommets distincts de G sont deux sous-ensembles de sommets de G' ayant une intersection vide. En conséquence de quoi les dérivées de deux composantes connexes d'un sous-graphe de G ne peuvent être connexes. □

1.4.9 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 :
- $S \rightarrow \mathcal{E}$, on a alors un **graphe pondéré sur les sommets**, aussi appelé **graphe à sommets valués**, 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

$$P(G) = \sum_{s \in S(G)} P(s); \tag{1.1}$$

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

$$P(G) = \sum_{a \in A(G)} P(a). \tag{1.2}$$

Remarque 1.25.
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 1.15, le poids du graphe reste alors inchangé. Voir les Figures 1.14a et 1.14b.
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 1.14c et 1.14d.

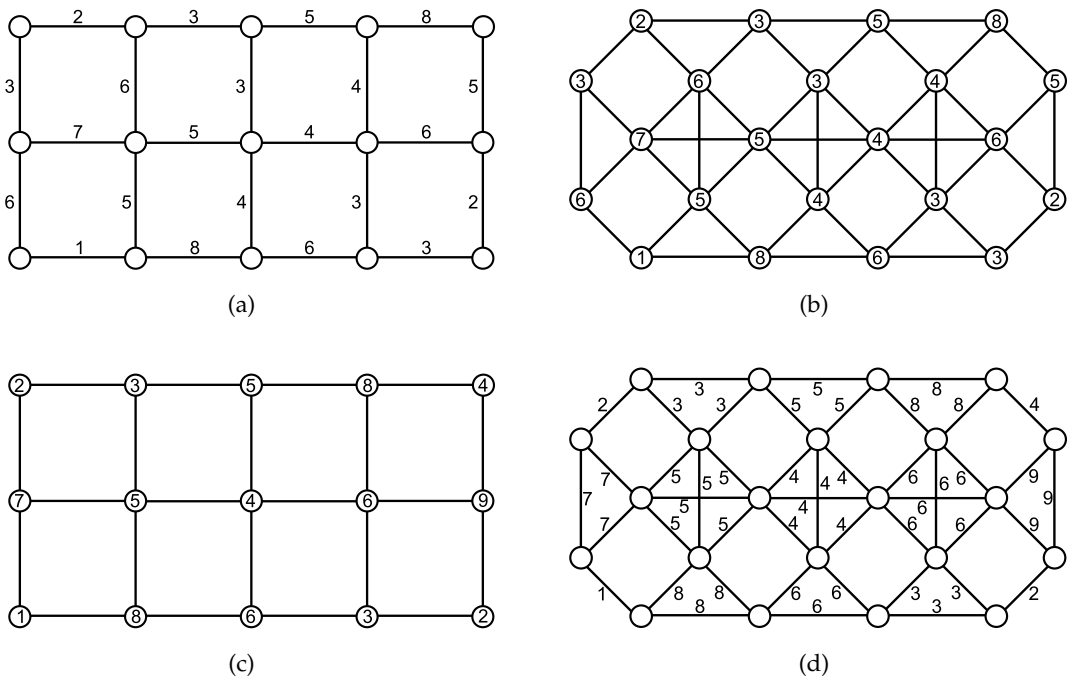


FIGURE 1.14 – (a) Graphe G_1 à arêtes valuées ; (b) Graphe G_1' dérivé de G_1 à sommets valués ; (c) Graphe G_2 à sommets valués ; (d) Graphe G_2' dérivé de G_2 à arêtes valuées.

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.

1.4.9.1 Descente et plus grande pente

Soit un graphe $G = (S, A)$ avec une application de poids P. Soit E l'ensemble d'éléments de G sur lequel s'effectue la pondération.

Soit un chemin (ou une chaîne) $\pi = \langle x_0, \ldots, x_\ell \rangle$ avec pour tout entier $k \in [0; \ell], x_k \in E$.

Le chemin (ou la chaîne) π est un **chemin descendant** (ou une **chaîne descendante**) (sur G, pour P) si pour tout entier $k \in [1; \ell], P(x_{k-1}) \geq P(x_k)$.

Dans un graphe à sommets valués, un chemin (ou une chaîne) de sommets $\pi = \langle s_0, \ldots, s_\ell \rangle$ est **de plus grande pente** (sur G, pour P) si π est un chemin (ou une chaîne) descendant(e) et si pour tout entier $k \in [1; \ell], P(s_k) = \min\{P(y) \mid y \in \Gamma(s_{k-1})\}$.

Dans un graphe à arcs valués (ou arêtes valuées), un chemin d'arcs (ou une chaîne d'arêtes) $\pi = \langle a_0, \ldots, a_\ell \rangle$ est **de plus grande pente** (sur G pour P) si π est un chemin (ou une chaîne) descendant(e) et si pour tout entier $k \in [1; \ell], P(a_k) = \min\{P(y) \mid y \in \gamma(a_{k-1})\}$.

1.4.9.2 Altitude d'un chemin et altitude de connexion

Soit un graphe $G = (S, A)$ avec une application de poids P. Soit E l'ensemble d'éléments de G sur lequel s'effectue la pondération.

Soit un chemin (ou une chaîne) $\pi = \langle x_0, \ldots, x_\ell \rangle$ avec pour tout entier $k \in [0; \ell], x_k \in E$.

L'**altitude**, du chemin (ou de la chaîne) π (pour P sur G), noté $H_P(\pi)$, ou plus simplement $H(\pi)$ lorsqu'il n'y a pas d'ambiguïté sur l'application de poids considérée, est l'altitude maximum des éléments de π (i.e. $H(\pi) = \max_{k \in [0;\ell]}\{P(x_k)\}$).

Soient x et y deux éléments de E. On note $\Pi(x, y)$ l'ensemble des chemins dans G allant de x à y.

Soient X et Y deux sous-graphes de G. On note $\Pi(X, Y)$ l'ensemble des chemins dans G allant de X à Y (i.e. $\Pi(X, Y) = \cup_{x \in X, y \in Y}\{\Pi(x, y)\}$).

L'**altitude de connexion**, ou la **valeur de connexion**, entre deux éléments x et y de E, noté $H_P(x, y)$, ou plus simplement $H(x, y)$ lorsqu'il n'y a pas d'ambiguïté sur l'application de poids considérée, est l'altitude la plus basse des altitudes de $\Pi(x, y)$ (i.e. $H_P(x, y) = \min_{\pi \in \Pi(x,y)}\{H_P(\pi)\}$).

De même, l'**altitude de connexion**, ou la **valeur de connexion**, entre deux sous-graphes X et Y de G, noté $H_P(X, Y)$, ou plus simplement $H(X, Y)$ lorsqu'il n'y a pas d'ambiguïté sur l'application de poids considérée, est l'altitude la plus basse des altitudes de $\Pi(X, Y)$ (i.e. $H_P(X, Y) = \min_{\pi \in \Pi(X,Y)}\{H_P(\pi)\}$).

1.4.9.3 Seuillage d'un graphe

Soit un graphe $G = (S, A)$ avec une application de poids P. Soit E l'ensemble d'éléments de G sur lequel s'effectue la pondération.

Pour tout $h \in \mathbb{R}$, on note $E_P^h = \{e \in E \mid P(e) \leq h\}$ et $E_P^{\overline{h}} = \{e \in E \mid P(e) \geq h\}$.

Soit X un sous-graphe de G.

On appelle **seuillage inférieur**, ou **section inférieure**, (de X, pour P) au niveau h le sous-graphe induit par $E(X) \cap E_P^h$, noté X_P^h. Quand il n'y a pas d'ambiguïté sur l'application de poids, on notera simplement X^h.

On appelle **seuillage supérieur**, ou **section supérieure**, (de X, pour P) au niveau h le sous-graphe induit par $E(X) \cap E_P^{\overline{h}}$, noté $X_P^{\overline{h}}$. Quand il n'y a pas d'ambiguïté sur l'application de poids, on notera simplement $X^{\overline{h}}$.

Remarque 1.26. *Pour tout couple $(h, h') \in \mathbb{R}^{+2}$ tel que $h < h'$, nous avons $X^{h'} \subseteq X^h$ et $X^{\overline{h}} \subseteq X^{\overline{h'}}$.*
Pour tout $h \in \mathbb{R}^{+2}$, généralement on a $X^h \neq \overline{X^{\overline{h}}}$.

Des exemples de seuillages de graphes non-orientés se trouvent dans les figures 1.15, 1.16, 1.17 et 1.18.

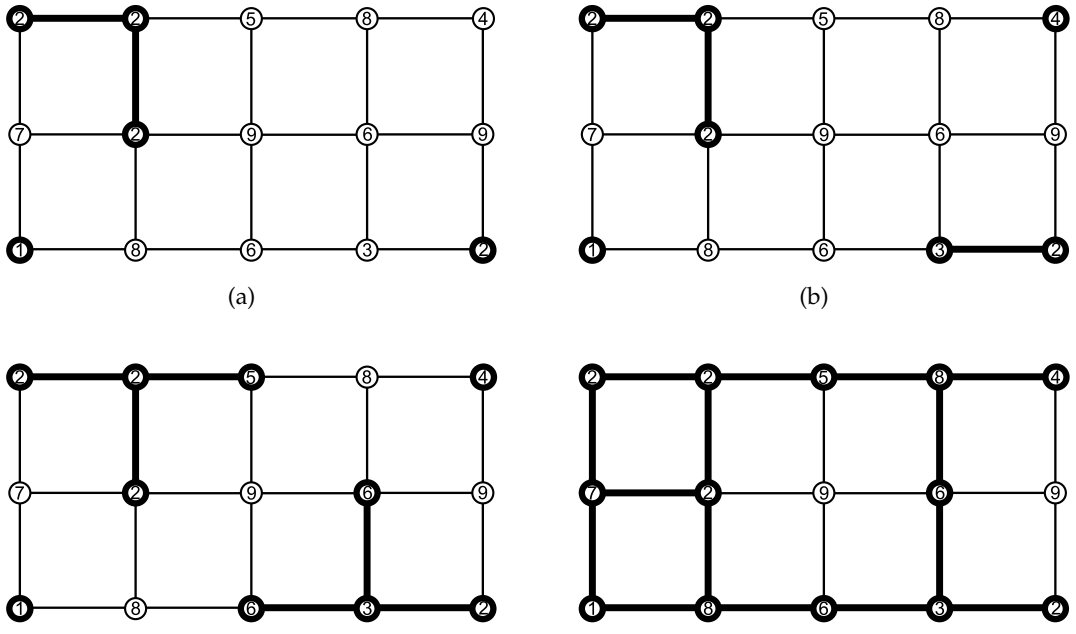


FIGURE 1.15 – Graphe G et application de poids P sur les sommets avec (en gras) : (a) G^2 ; (b) G^4 ; (c) G^6 ; (d) G^8.

1.4.9.4 Plateau

Soit un graphe $G = (S, A)$ avec une application de poids P. Soit E l'ensemble d'éléments de G sur lequel s'effectue la pondération.

Le sous-graphe X de G est un **plateau** de P si :

- X est connexe,
- tous les éléments de $E(X)$ ont la même altitude (le même poids), auquel on fera référence en tant que **altitude de** X, et

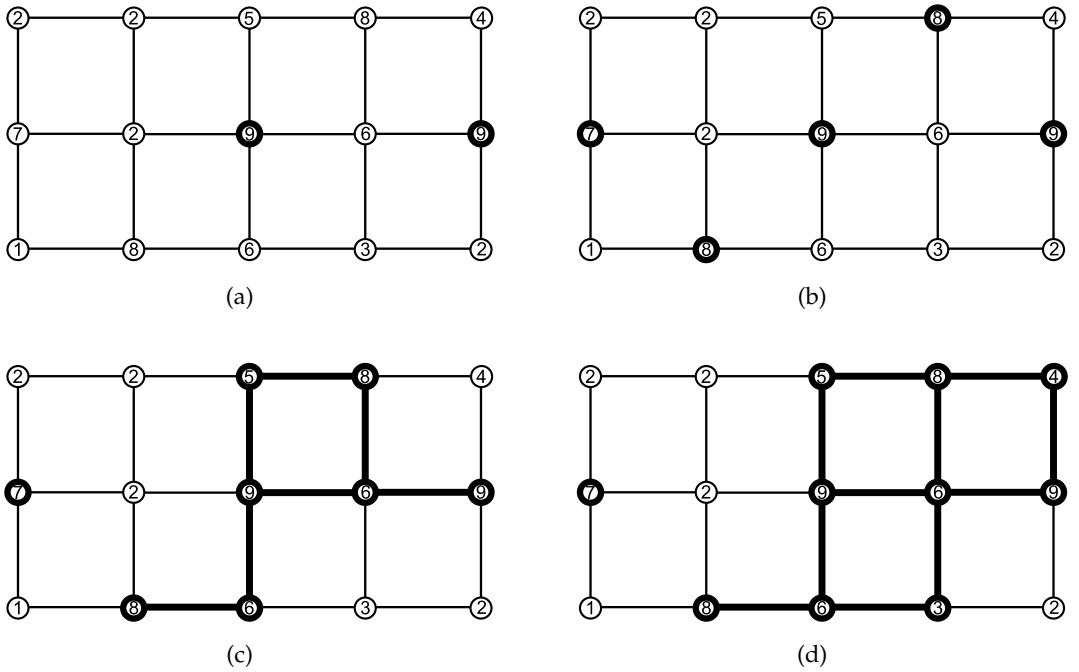


FIGURE 1.16 – Graphe G et application de poids P sur les sommets avec (en gras) : (a) $G^{\overline{9}}$; (b) $G^{\overline{7}}$; (c) $G^{\overline{5}}$; (d) $G^{\overline{3}}$.

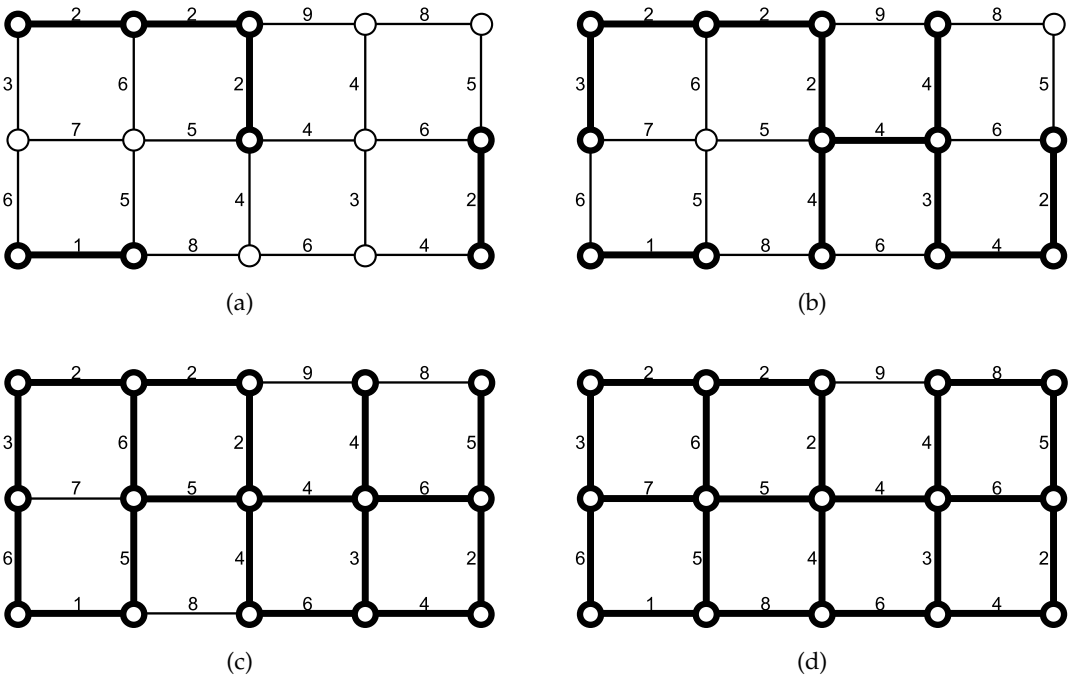


FIGURE 1.17 – Graphe G et application de poids P sur les arêtes avec (en gras) : (a) G^2 ; (b) G^4 ; (c) G^6 ; (d) G^8.

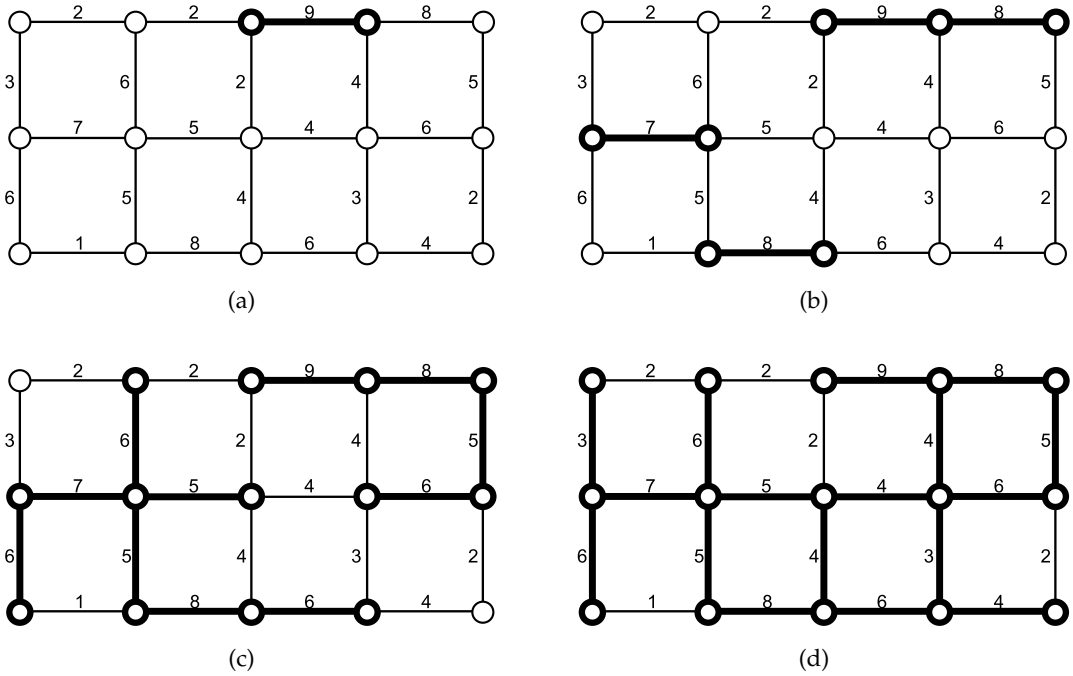


FIGURE 1.18 – Graphe G et application de poids P sur les arêtes avec (en gras) : (a) $G^{\overline{9}}$; (b) $G^{\overline{7}}$; (c) $G^{\overline{5}}$; (d) $G^{\overline{3}}$.

– X est maximal pour les conditions précédentes, autrement dit il n'existe pas de sous-graphe connexe de G qui l'inclut strictement.

Des exemples de cette définition sur des graphes non-orientés se trouvent dans les figures 1.19 et 1.20.

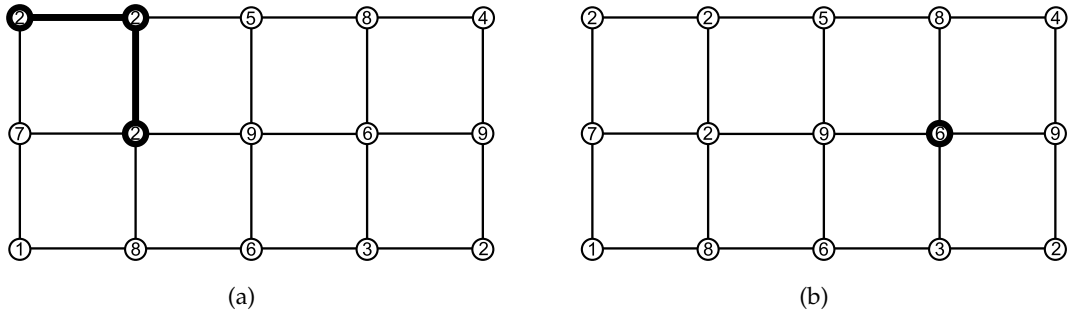


FIGURE 1.19 – Graphe G et application de poids P sur les sommets avec deux exemples de plateaux (en gras).

1.4.9.5 Minima et maxima régionaux d'un graphe pondéré

Soit un graphe $G = (S, A)$ avec une application de poids P. Soit E l'ensemble d'éléments de G sur lequel s'effectue la pondération.

Le sous-graphe X de G est un **minimum (régional)** de P si :

– X est un plateau, et

– l'altitude de tout élément de E adjacent à X est strictement supérieur à l'altitude de X.

Le sous-graphe X de G est un **maximum (régional)** de P si :

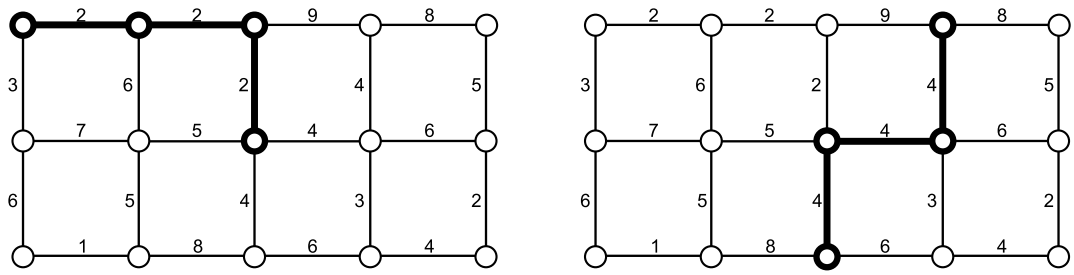


FIGURE 1.20 – Graphe G et application de poids P sur les arêtes avec deux exemples de plateaux (en gras).

- X est un plateau, et
- l'altitude de tout élément de E adjacent à X est strictement inférieur à l'altitude de X.

En d'autres termes, un minimum régional est une composante de de G^h, avec $h \in \mathbb{R}$, qui ne contient pas de composante connexe de $G^{h'}$, avec $h \in \mathbb{R}$ et $h' < h$. Respectivement, un maximum régional est une composante de de G^h, avec $h \in \mathbb{R}$, qui ne contient pas de composante connexe de $G^{h'}$, avec $h \in \mathbb{R}$ et $h' > h$.

On définit par $Min(P)$ l'union de tous les sous-graphes de G qui sont des minima régionaux de G. Respectivement, on définit par $Max(P)$ l'union de tous les sous-graphes de G qui sont des maxima régionaux de G.

Des exemples de ces définitions sur des graphes non-orientés se trouvent dans les figures 1.21 et 1.22.

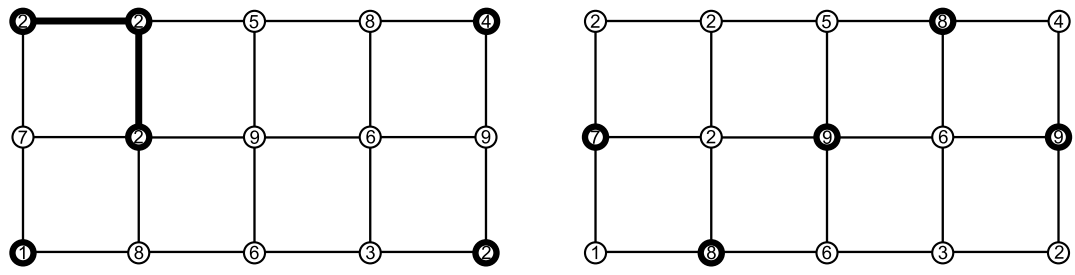


FIGURE 1.21 – Graphe G et application de poids P sur les sommets avec (en gras) : (a) $Min(P)$; (b) $Max(P)$.

1.5 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.

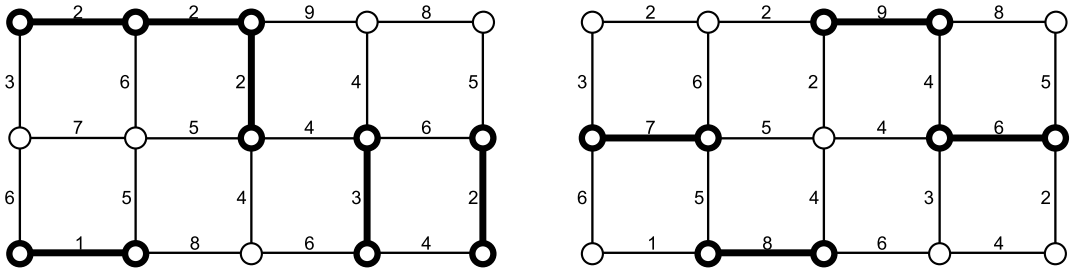


FIGURE 1.22 – Graphe G et application de poids P sur les arêtes avec (en gras) : (a) $Min(P)$; (b) $Max(P)$.

1.5.1 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 $[O; d_1] \times [0; d_2] \times \ldots \times [0; d_n]$ 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).

1.5.2 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 considére 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

$$\sqrt{(x_1 - y_1) + (x_2 - y_2)} \leq \sqrt{1} = 1 \tag{1.3}$$

(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

$$\sqrt{(x_1 - y_1) + (x_2 - y_2)} \leq \sqrt{2} \tag{1.4}$$

(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

$$\sqrt{(x_1 - y_1) + (x_2 - y_2) + (x_3 - y_3)} \leq \sqrt{1} = 1 \tag{1.5}$$

(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

$$\sqrt{(x_1 - y_1) + (x_2 - y_2) + (x_3 - y_3)} \leq \sqrt{2} \tag{1.6}$$

(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

$$\sqrt{(x_1 - y_1) + (x_2 - y_2) + (x_3 - y_3)} \leq \sqrt{3} \tag{1.7}$$

(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 n**D** (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

$$\sqrt{(x_1 - y_1) + (x_2 - y_2) + \ldots + (x_n - y_n)} \leq \sqrt{d} \tag{1.8}$$

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

Exemple 1.27.
Si l'on considère l'image 2D de taille 3×3 *de la figure 1.23 associée au 4-voisinage, alors le pixel* e *a pour voisins les pixels* $\{b, d, f, h\}$. *Par contre, si elle est associée au 8-voisinage, alors le pixel* e *a pour voisins les pixels* $\{a, b, c, d, f, g, h, i\}$.

1.5.3 Image, voisinage et graphe

La définition de voisinnage 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 voisinnages 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 :

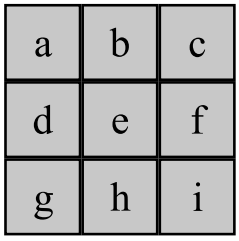


FIGURE 1.23 – Image 2D de taille 3×3 (3 pixels par 3 pixels).

- 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.

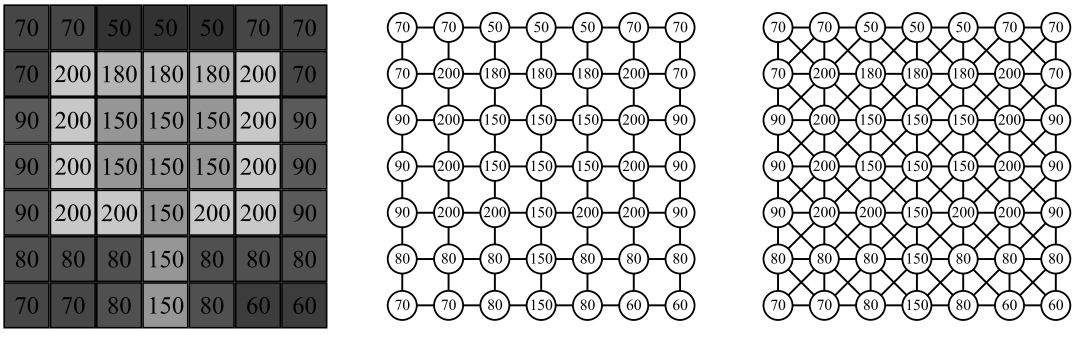
On appelle **grille (de dimension n)** tout graphe (S, A) (orienté ou non) tel que $S \subseteq \mathbb{Z}^n$. Typiquement, on a $S = [O; d_1] \times [0; d_2] \times \ldots \times [0; d_n]$.

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 1.24.



(a) (b) (c)

FIGURE 1.24 – (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.

1.6 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. Nous étudierons davantage de lien entre image et graphe dans le chapitre suivant à travers la présentation de différents concepts de segmentation de graphe.

Segmentation et partition de graphes

2

Avant de voir, dans les chapitres suivants, comment segmenter ou partitionner un graphe et quels algorithmes peuvent être employés dans ce but, nous définissons ici ce que nous entendons plus précisément par segmentation de graphes et sous quelles formes peut se présenter un tel résultat. La réprésentation de la segmentation de graphe que nous utilisons dans ce manuscrit est un ensemble d'éléments, sommets ou arêtes, servant de délimitation, de frontière, aux différentes régions constituant la dite segmentation. Nous abordons trois définitions de tels ensembles : les coupes, les ensembles frontières de sommets et les ensembles frontières d'arêtes. Nous présentons ensuite des liens existant entre elles ainsi que certaines de leurs propriétés. Enfin, nous discutons sur le choix de tel ou tel type de résultat selon l'application désirée.

Bien qu'applicables sans trop de difficultés aux graphes orientés, les notions abordées dans ce chapitre sont toutes considérées dans le cas de graphes non-orientés uniquement. C'est pourquoi, dans la suite de ce chapitre, nous considérons le graphe non-orienté $G = (S, A)$.

De plus, ici encore, quand cela s'avére utile, nous notons E l'ensemble d'éléments traité. Celui-ci peut indistinctement être l'ensemble de sommets (i.e. $E = S$) ou d'arêtes (i.e. $E = A$) de G (comme déjà introduit dans la section 1.4.9 pour le type d'élément pondéré dans le graphe). Ceci nous permet ainsi, entre autres, d'unifier les définitions d'ensembles frontières de sommets et d'arêtes.

Sommaire du chapitre

Figures du chapitre

2.1 Extension, extension couvrante, extension maximale et coupe

La notion d'extension fut introduite pour la première fois dans le cadre de graphes connexes dans [26]. Nous avons ensuite étendu cette notion pour le cas de graphes quelconques dans [7, 8] avec les définitions ci-après.

Définition 2.1 (Extensions et coupe)**.**
Soient $G_1, G_2, \ldots, G_n$ les différentes composantes connexes de $G = (S, A)$. Soient M et X deux sous-graphes de G. Soient également, pour tout $i \in \{1, 2, \ldots, n\}$, $M_i = M \cap G_i$ et $X_i = X \cap G_i$.

- *On dit que X est une **extension** de M (sur G) si, pour tout $i \in \{1, 2, \ldots, n\}$, $M_i \subseteq X_i$ et si chaque composante connexe de X_i contient exactement une composante connexe de M_i.*
- *On dit que X est une **extension couvrante** de M (sur G) si X est une extension de M et si $S(X) = S$.*
- *On dit que X est une **extension maximale** de M (sur G) si X est une extension de M et s'il n'y a pas d'extension de M qui contient strictement X.*
- *Soit $C \subseteq A$, on dit que C est une **coupe** relative à M (sur G) si le graphe induit par $\overline{C}$ est une extension maximale de M.*

Des exemples de ces définitions se trouvent dans la figure 2.1.

Remarque 2.2.
Soit M un sous-graphe de G.

- *Une extension maximale de M est toujours une extension couvrante de M.*
- *Pour toute extension X de M, il existe une extension maximale Y de M telle que $X \subseteq Y$.*
- *Pour tout $i \in [1, n]$, si $M_i = G_\emptyset$, alors n'importe quel sous-graphe connexe de G_i est une extension de M_i et G_i est l'extension maximale de X_i.*
- *Il est à noter que si aucune composante connexe de G ne contient plus d'une composante connexe de M, alors l'unique coupe relative à M est l'ensemble vide.*

Pour toute extension couvrante X de M, il existe une unique coupe C relative à X que l'on appelle **coupe induite** par X. La coupe C est également relative à M.

Remarque 2.3.
Les sommets qui sont les extrémités d'une arête comprise dans une coupe C appartiennent à deux composantes connexes différentes du graphe $(S, \overline{C})$.

2.2 Ensemble séparant et frontière

2.2.1 Graphe dual et ensemble séparant

Soit un ensemble de sommets ou d'arêtes $K \subseteq E$ de G. On appelle **graphe dual** à K, noté $\mathcal{D}(K)$, le sous-graphe induit par $\overline{K}$.

Des exemples de cette définition se trouvent dans la figure 2.2.

Remarque 2.4.
Par définition, le graphe dual d'une coupe est l'extension maximale qui induit celle-ci.

Dans un graphe, un **ensemble séparant** est un ensemble de sommets ou d'arêtes $K \subseteq E$ de G tel que :

- chaque composante connexe de G contient au moins une composante connexe du graphe dual à K (en notant $G_1, G_2, \ldots, G_n$ les différentes composantes connexes de G, on a alors $\forall i \in [1; n], |\mathcal{C}(G_i \cap \mathcal{D}(K))| \geq 1$), et
- le nombre total de composantes connexes de $\mathcal{D}(K)$ est strictement supérieur au nombre de composantes connexes de G (i.e. $|\mathcal{C}(\mathcal{D}(K))| \geq |\mathcal{C}(G)|$).

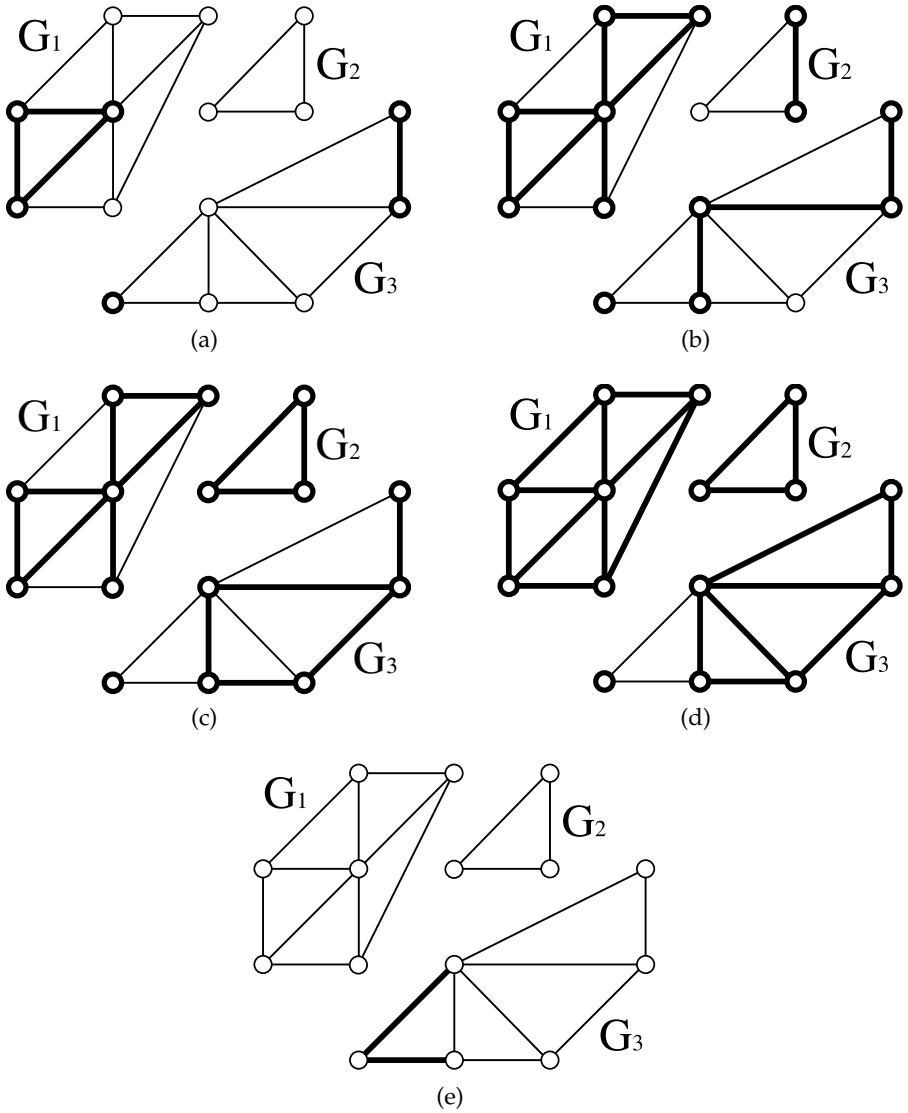


FIGURE 2.1 – Graphe G, composé de trois composantes connexes (G_1, G_2 et G_3), avec (en gras) : (a) un sous-graphe M ; (b) une extension relative à M ; (c) une extension couvrante relative à M ; (d) une extension maximale relative à M ; (e) une coupe relative à M.

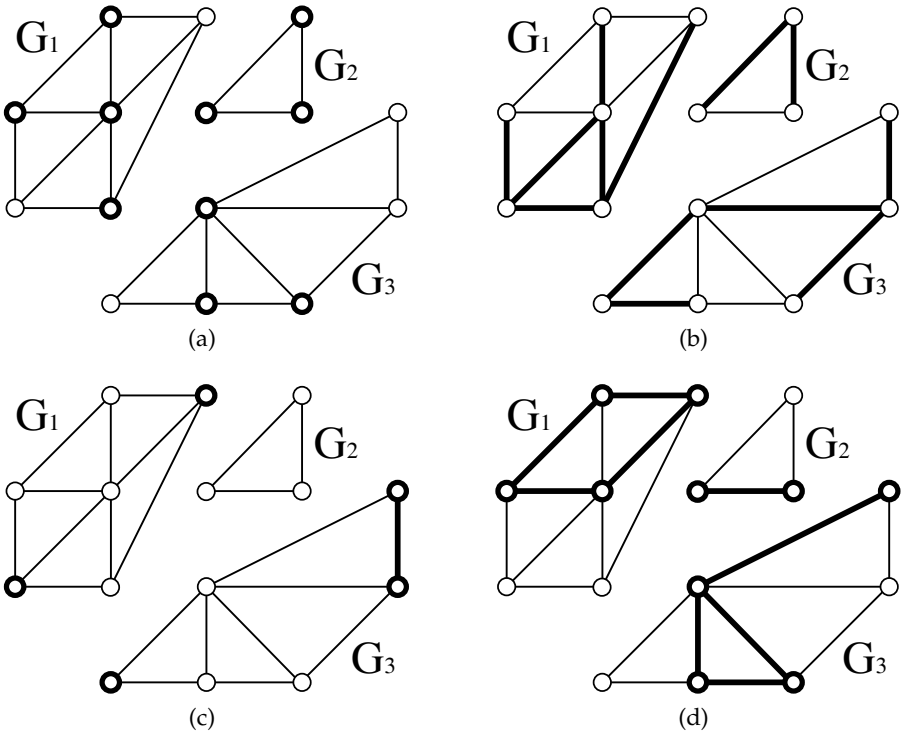


FIGURE 2.2 – Graphe G avec (en gras) : (a) un ensemble de sommets $K_S \in S$; (b) un ensemble d'arêtes $K_A \in A$; (c) $\mathcal{D}(K_S)$; (d) $\mathcal{D}(K_A)$.

On dit qu'un ensemble séparant K est relatif à un sous-graphe X de G si $\mathcal{D}(K)$ est une extension de X.

Des exemples de ces définitions se trouvent dans la figure 2.3. Les ensembles des figures 2.2a et 2.2b ne sont pas des ensembles séparants.

Remarque 2.5.
Il est à noter qu'une coupe n'est pas nécessairement un ensemble séparant.
Par exemple, la coupe de la figure 2.1e n'est pas un ensemble séparant.

2.2.2 Amincissement et ensemble frontière

Soient $K \subseteq E$ et $K' \subseteq E$ deux sous-ensembles de sommets ou d'arêtes de G. On dit que K' est un **amincissement** de K si :
- $K' = K$, ou
- il existe un amincissement K'' de K et un élément $e \in K''$ tels que :
  - $K' = K'' \setminus \{e\}$, et
  - e est adjacent à une unique composante connexe du graphe dual à K'' (i.e. $\mathcal{D}(K'')$) ou bien $e \in E(G_i)$ où G_i est une composante connexe de G telle que $E(G_i) \subseteq K''$.

Un amincissement d'un ensemble K peut ainsi être obtenu par suppression itérative d'éléments qui sont adjacents à une unique composante connexe de $\mathcal{D}(K)$ (ou en en supprimant un premier, quelconque, d'une composante connexe dont tous les éléments sont inclus dans K).

Des exemples de cette définition se trouvent dans la figure 2.4.

On déduit aisément des définitions d'amincissement et d'extension les propriétés suivantes.

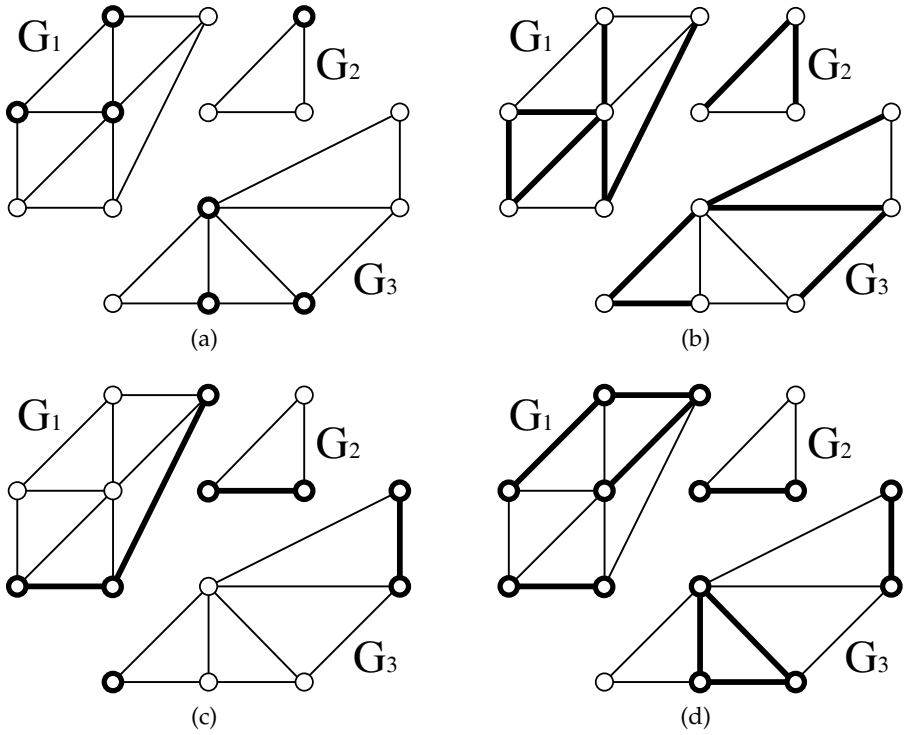


FIGURE 2.3 – Graphe G avec (en gras) : (a) un ensemble séparant de sommets $S' \in S$; (b) un ensemble séparant d'arêtes $A' \in A$; (c) $\mathcal{D}(S')$; (d) $\mathcal{D}(A')$.

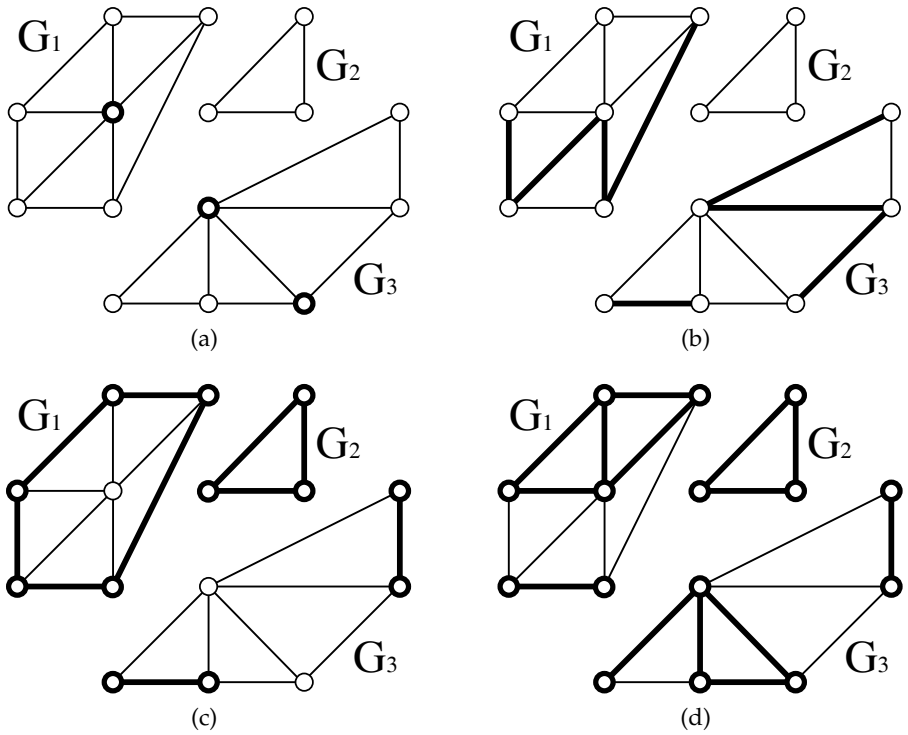


FIGURE 2.4 – Graphe G avec (en gras) : (a) un ensemble de sommets $S'' \in S$ qui est un amincissement de S' de la figure 2.3a ; (b) un ensemble d'arêtes $A'' \in A$ qui est un amincissement de A' de la figure 2.3b ; (c) $\mathcal{D}(S'')$; (d) $\mathcal{D}(A'')$.

Propriété 2.6.
Soit un ensemble d'arêtes $A' \in A$.
L'ensemble d'arêtes A'' est un amincissement de A' si et seulement si $\mathcal{D}(A'')$ est une extension de $\mathcal{D}(A')$.

Il n'y a pas de propriété d'équivalence similaire en ce qui concerne les ensembles de sommets mais l'implication de la propriété suivante.

Propriété 2.7.
Soit un ensemble de sommets $S' \in S$.
Si l'ensemble de sommets S'' est un amincissement de S', alors $\mathcal{D}(S'')$ est une extension de $\mathcal{D}(S')$.

Des exemples de ces propriétés se trouvent dans la figure 2.4 où $\mathcal{D}(S'')$ est une extension de $\mathcal{D}(S')$ de la figure 2.2c et $\mathcal{D}(A'')$ est une extension de $\mathcal{D}(A')$ de la figure 2.2d.

Remarque 2.8.
Il est à noter que la réciproque du théorème 2.7 n'est, en général, pas vraie.
Par exemple, le sous-graphe de la figure 2.1d est une extension de $\mathcal{D}(S')$ de la figure 2.3c mais il n'est le graphe dual d'aucun ensemble séparant de sommets.

Un **ensemble frontière** est un ensemble séparant qui est minimum, c'est-à-dire dont aucun élément ne peut être retiré sans modifier le nombre de composantes connexes de son graphe dual. Autrement dit, il n'existe aucun amincissement d'un ensemble frontière distinct de celui-ci.
Par définition, tout ensemble séparant contient au moins un ensemble frontière relatif au même graphe.

Des exemples de cette définition se trouvent dans la figure 2.5.

On déduit aisément des définitions d'ensembles séparants et d'ensembles frontières la propriété suivante.

Propriété 2.9.
Un ensemble séparant de sommets ou d'arêtes K de G est un ensemble frontière de G si et seulement si aucun élément de K n'est adjacent à une unique composante connexe de son graphe dual $\mathcal{D}(K)$.

Les ensembles frontières prennent tout leur intérêt lorsque l'on souhaite segmenter un graphe. Lorsque l'on parle de segmentation, cela évoque la recherche d'une partition des éléments considérés. En l'occurrence, dans le cadre de graphes les éléments en question sont, dans la majorité des cas, les sommets. Ainsi, les sommets d'une même composante connexe du graphe dual à l'ensemble frontière sont tous regroupés dans une même région du graphe segmenté.

Remarque 2.10.
Il est à noter qu'un ensemble frontière de sommets ne permet pas d'obtenir une partition des sommets du graphe puisque les sommets de la dite frontière ne sont pas étiquetés comme appartenant à l'une ou l'autre des régions obtenues (étant donné qu'ils n'appartiennent à aucune composante connexe du graphe dual).
De plus, un ensemble frontière de sommets ne permet pas non plus de créer une partition d'arêtes puisque certaines arêtes peuvent très bien avoir pour extrémités deux sommets de l'ensemble frontière de sommets.

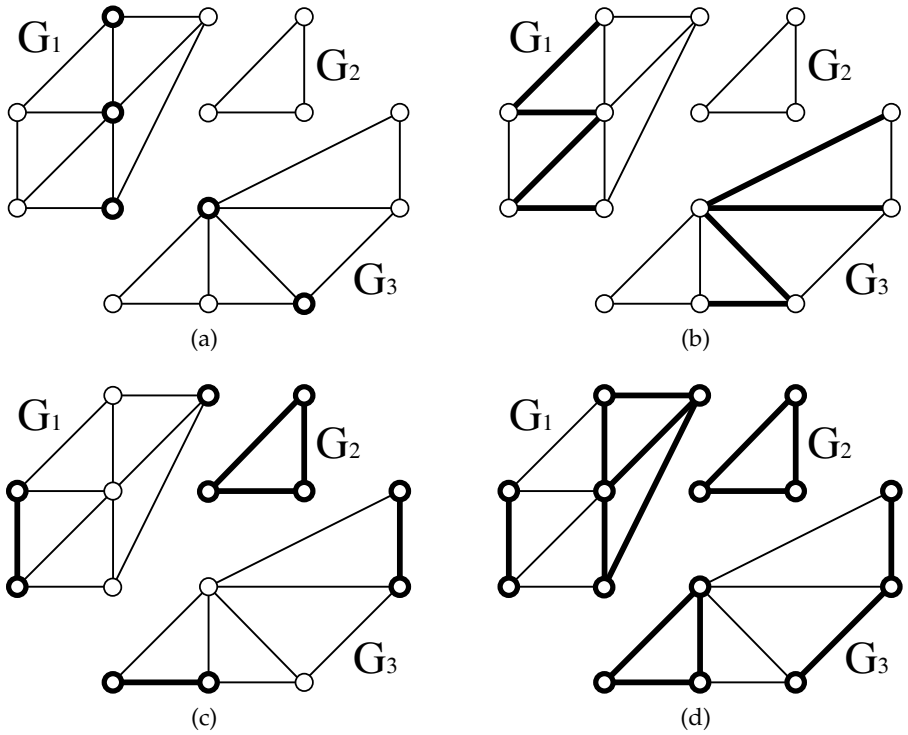


FIGURE 2.5 – Graphe G avec (en gras) : (a) un ensemble frontière de sommets $S' \in S$; (b) un ensemble frontière d'arêtes $A' \in A$; (c) $\mathcal{D}(S')$; (d) $\mathcal{D}(A')$.

2.2.3 Minceur et finesse

Dans un sous-ensemble d'éléments $E' \subseteq E$, un élément $e \in E'$ est dit **intérieur** à E' si tous les éléments qui lui sont adjacents sont dans E'. Autrement dit, e est intérieur à E' s'il n'est pas adjacent au graphe dual à E'. Sinon, on dit que e est un élément de **bord** de E'.

Un ensemble d'éléments est dit **mince** (pour G) si aucun élément qui le compose n'est intérieur.

Des exemples de cette définition se trouvent dans la figure 2.6 où les ensembles représentés sur 2.6b, 2.6c et 2.6d, respectivement 2.6f et 2.6g et 2.6d, sont des amincissements de l'ensemble représenté sur 2.6a, respectivement 2.6e.

Remarque 2.11.
Il est à noter que la remarque 2.10 reste valable même pour les ensembles frontières de sommets qui sont minces.

Il est à souligner qu'il n'existe pas nécessairement d'amincissement non distinct à un ensemble séparant qui n'est pas mince (et qui, par définition, est un ensemble frontière - voir figure 2.6c par exemple). Cependant, il existe des ensembles séparants minces qui possèdent un amincissement distinct (et qui, par définition, ne sont donc pas des ensembles frontières - voir figures 2.6b et 2.6f par exemple). De par ces constatations, la définition d'ensemble mince ne semble pas être suffisante pour caractériser les différents types d'ensembles séparants. En conséquence de quoi nous introduisons la définition qui suit.

Un ensemble d'éléments est dit **fin** (pour G) si chacun de ses éléments est adjacent à au moins deux composantes connexes de son graphe dual.

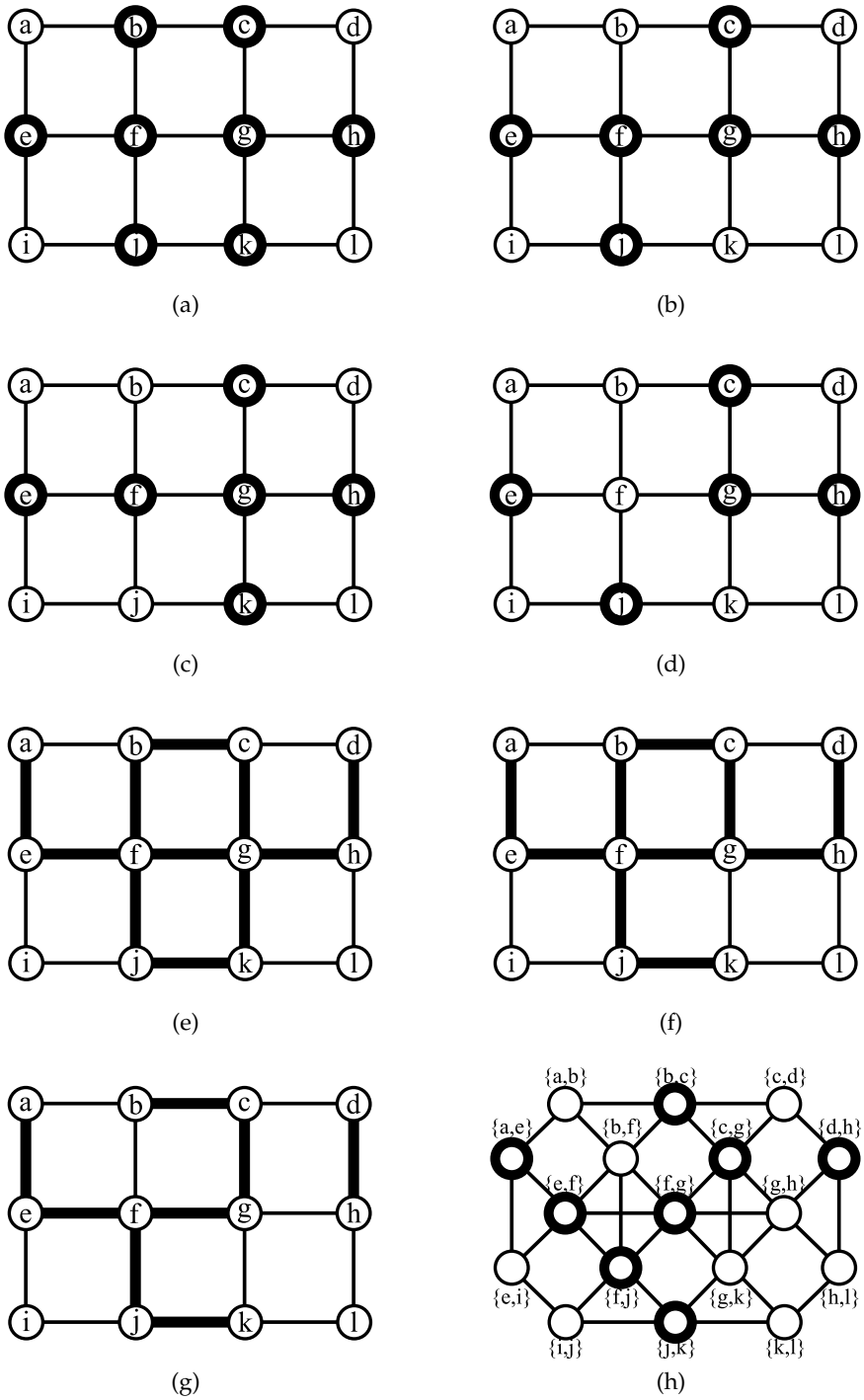


FIGURE 2.6 – Graphe G avec (en gras) : (a) un ensemble séparant de sommets ; (b) un ensemble séparant mince de sommets ; (c) un ensemble frontière non-mince de sommets ; (d) un ensemble frontière mince de sommets ; (e) un ensemble séparant d'arêtes ; (f) un ensemble séparant mince et fin d'arêtes ; (g) un ensemble frontière (mince et fin) d'arêtes. (h) Graphe G' dérivé de G avec (en gras) l'ensemble frontière mince et fin de sommets dérivants de l'ensemble frontière d'arêtes de la figure 2.6g.

Cette définition est plus forte que celle d'ensemble mince comme le montre la propriété 2.12, aisément déduite des définitions d'ensembles minces et d'ensembles fins.

Propriété 2.12.
Si l'ensemble $K \subseteq E$ de G est fin, alors K est mince.

La réciproque de la propriété 2.12 n'est généralement pas vraie comme le montrent les figures 2.6b et 2.6f, qui représentent un ensemble séparant de sommets, respectivement d'arêtes, qui est mince mais pas fin.

On remarque que la définition d'ensemble fin rejoint celle d'ensemble frontière puisque la définition d'ensemble fin fait appel au nombre de composantes connexes du graphe dual auquel chacun de ses éléments est adjacent, tout comme la propriété 2.9 sur les ensembles frontières. Ceci est mis en évidence à travers la propriété 2.13.

Propriété 2.13.
Un ensemble $K \subseteq E$ de G est fin si et seulement si K est un ensemble frontière mince de G.

Remarque 2.14.
Il est à noter que la condition de minceur dans la propriété 2.13 est nécessaire. En effet, sans cela, un ensemble frontière de sommets n'est pas nécessairement fin, comme le montre la figure 2.6c. Nous verrons dans le théorème 2.15 que ce problème ne se pose pas pour les ensembles frontières d'arêtes.

Des exemples d'ensembles séparants fins, qui sont donc des ensembles frontières minces, se trouvent dans les figures 2.6d et 2.6g.

La propriété de "finesse" est souvent souhaitée dans la segmentation de graphes de sorte à maximiser le nombre d'éléments étiquetés dans les différentes régions mais qui peut aussi s'avérer nécessaire pour certains traitements a posteriori, comme la fusion de régions par exemple. Le théorème suivant montre que c'est toujours le cas lorsque l'on travaille avec des ensembles frontières d'arêtes.

Théorème 2.15.
Tout ensemble frontière d'arêtes est fin.

Le théorème 2.15 nous permet ainsi de déduire aisément des propriétés 2.12 et 2.13 le corollaire suivant.

Corollaire 2.16.
Un ensemble séparant d'arêtes $K \subseteq A$ de G est fin si et seulement si K est un ensemble frontière de G.

La remarque 2.11 et le théorème 2.15 mettent ainsi clairement en évidence l'avantage que procure l'utilisation d'ensembles frontières d'arêtes par rapport à celui d'ensembles frontières de sommets dans le cadre de graphes quelconques.

2.3 Lien entre ensemble frontière d'arêtes et coupe

Comme présenté en début de chapitre, les notions d'ensembles frontières d'arêtes et de coupes servent toutes deux à segmenter un graphe. Bien que de définitions différentes, nous verrons dans cette section les liens qui existent entre elles. En effet, des similitudes existent entre elles puisque toutes deux sont des ensembles d'arêtes et utilisent les extensions dans leur définition. Ceci est mis en évidence par la propriété 2.17.

Propriété 2.17.
Tout ensemble frontière d'arêtes est une coupe.

Remarque 2.18.
Il est à noter que la réciproque du corollaire 2.17 n'est, en général, pas vraie. En effet, une coupe n'est pas nécessairement fine ou même mince, selon les définitions données plus haut, et peut ainsi contenir des arêtes intérieures dans le cas où l'extension couvrante qui l'induit contient des sommets isolés adjacents dans G.
De plus, si l'on considère un ensemble frontière d'arêtes $E_F \subseteq A$ et une coupe $C \subseteq A$, alors, par définition, le graphe $(S, \overline{C})$ peut contenir des sommets isolés alors que le graphe $(S, \overline{E_F})$ ne le peut pas. C'est ce que suggérait la remarque 2.5 en montrant une coupe qui n'est pas un ensemble frontière (voir figure 2.1e).

La différence entre ensemble frontière d'arêtes et coupe soulevée dans la remarque 2.18 s'explique par le fait qu'un ensemble frontière d'arêtes est défini par son graphe dual alors qu'une coupe est définie par l'extension du sous-graphe quelconque auquel elle est relative. Par définition d'un graphe dual à un ensemble d'arêtes, chacune de ses composantes connexes contient au moins une arête, ce qui n'est pas le cas pour un sous-graphe quelconque. En conséquence de quoi il ne peut y avoir de sommets isolés segmentés par un ensemble frontière d'arêtes alors qu'une coupe le peut.

Des exemples de cette remarque se trouvent dans la figure 2.7.

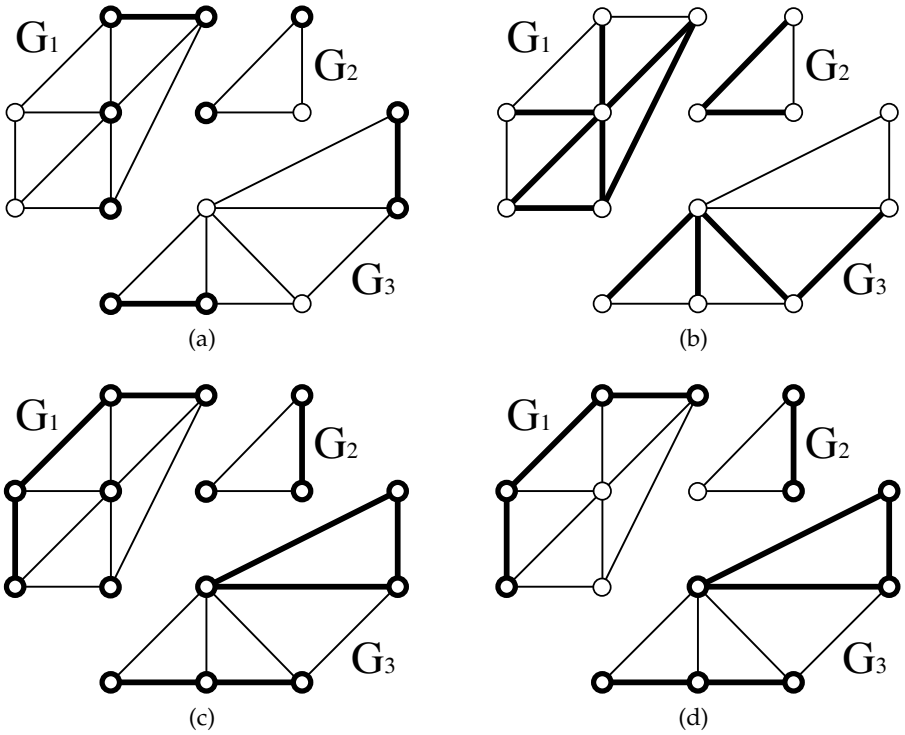


FIGURE 2.7 – Graphe G, composé de trois composantes connexes (G_1, G_2 et G_3), avec (en gras) : (a) un sous-graphe M ; (b) une coupe C relative à M qui n'est pas mince ; (c) $(S, \overline{C})$ (contenant trois sommets isolés) ; (d) $\mathcal{D}(C)$ (ne contenant pas de sommet isolé).

La remarque 2.3, selon laquelle toute arête d'une coupe C est adjacente à exactement deux composantes connexes du graphe $(S, \overline{C})$ qui l'induit, est en étroite relation avec le théorème 2.15, dont on peut déduire que toute arête d'un ensemble frontière d'arêtes K est adjacente à exactement deux composantes connexes du graphe dual à K. Toutefois, une

coupe n'est pas nécessairement fine puisqu'elle n'est pas toujours un ensemble frontière (voir les exemples des figures 2.1e et 2.7b).

2.4 Segmentation de graphe et graphe dérivé

Dans cette section, nous mettrons en évidence les propriétés liées à la dérivation de graphes segmentés. Nous commencerons par voir une propriété des ensembles séparants lors de la dérivation de graphes. Nous verrons ensuite que cette propriété ne peut être étendue aux ensembles frontières et aux ensembles minces ou fins que lorsque ceux-ci sont des ensembles d'arêtes. Enfin, nous verrons quelles conséquences ont ces propriétés lors de la dérivation de coupes.

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

Propriété 2.19.
Soit $K \subset E$ un ensemble séparant (de sommets ou d'arêtes) de G et $K' = \partial(K)$ l'ensemble dérivé de K (si K est un ensemble de sommets, alors K' est un ensemble d'arêtes, et réciproquement). L'ensemble d'éléments K est un ensemble séparant de G si et seulement si K' est un ensemble séparant de G'.

Preuve de la propriété 2.19.
Par construction, on sait que le graphe dual à $K' = \partial(K)$ est égal au dérivé du graphe dual à K (i.e. $\mathcal{D}(\partial(K)) = \partial(\mathcal{D}(K))$). Or, d'après la proposition 1.24, nous savons que $|\mathcal{C}(\partial(\mathcal{D}(K)))| = |\mathcal{C}(\mathcal{D}(K))|$. En conséquence de quoi, K' est donc bien un ensemble séparant. □

Propriété 2.20.
L'ensemble d'arêtes $K \subseteq A$ est un ensemble frontière d'arêtes de G si et seulement si $\partial(K)$ est un ensemble frontière de sommets de G'.

Preuve de la propriété 2.20.
D'après la propriété 2.9 nous savons qu'un ensemble séparant qui est un ensemble frontière ne possède aucun élément qui soit adjacent à une unique composante connexe de son graphe dual. D'après la remarque 1.21, la dérivation d'arêtes conserve les propriétés d'adjacence. On déduit donc de la propriété 2.19 qu'un ensemble séparant d'arêtes est un ensemble frontière d'arêtes de G si et seulement si le dérivé de cet ensemble est un ensemble frontière de sommets de G'. □

Remarque 2.21.
Il est à noter qu'il n'y a pas de résultat équivalent à celui de la propriété 2.20 pour un ensemble frontière de sommets. En effet, si l'on considère la dérivée d'un ensemble frontière de sommets, de par le fait que la dérivée d'un sommet est un ensemble d'arêtes, la propriété 2.19 nous permet d'affirmer qu'il s'agit d'un ensemble séparant de sommets, mais, la plupart du temps, il ne s'agit pas d'un ensemble frontière.

Les deux propriétés suivantes, traitant des notions de minceur et de finesse dans les graphes dérivés, peuvent être facilement déduite de la remarque 1.21 sur la conservation des propriétés d'adjacence lors de la dérivation.

Propriété 2.22.
L'ensemble d'arêtes $K \subseteq A$ est mince pour G si et seulement si $\partial(K)$ est un ensemble de sommets mince pour G'.

Propriété 2.23.
L'ensemble séparant d'arêtes $K \subseteq A$ est fin pour G si et seulement si $\partial(K)$ est un ensemble séparant de sommets fin pour G'.

Les propriétés montrées jusqu'ici nous permettent de déduire aisément le corollaire 2.24.

Corollaire 2.24.
L'ensemble d'arêtes $K \subseteq A$ est un ensemble frontière (fin) d'arêtes de G si et seulement si $\partial(K)$ est un ensemble frontière fin de sommets de G'.

Un exemple de ce corollaire se trouve dans les figures 2.6g et 2.6h.

Remarque 2.25.
Il est à noter que la dérivée d'une coupe qui n'est pas un ensemble frontière d'arêtes ne donne pas un ensemble frontière de sommets. En effet, les sommets isolés de l'extension couvrante qui induit cette coupe n'ont pas de sommets pour les représenter dans le graphe dérivé. En conséquence de quoi l'ensemble de sommets qui en est le dérivé ne sera pas mince. Tout au plus, cet ensemble pourra être un ensemble séparant de sommets du graphe dérivé.

Grâce au théorème 2.15, on peut déduire du corollaire 2.24 le théorème sur les ensembles frontières de sommets dans les graphes dérivés qui suit.

Théorème 2.26.
Dans un graphe dérivé, tout ensemble frontière de sommets est fin.

Un exemple d'ensemble frontière fin de sommets dans un graphe dérivé se trouve dans la figure 2.6h.

Les théorèmes et propriétés exposés ci-dessus mettent en évidence certains liens qui existent entre les ensembles frontières de sommets ou d'arêtes à travers les graphes dérivés. La section suivante introduira quelques cadres particuliers qui sont habituellement utilisés lors de la segmentation de graphes et nous y verrons que la dérivation de graphes permet de lier certains de ces cadres.

2.5 Graphes usuels pour la segmentation

Le cadre des graphes est souvent employé pour la segmentation d'images. En effet, comme on l'a vu dans la section 1.5, une image peut être équipée de notions topologiques grâce à son association à un type de voisinage qui en fait dès lors un graphe, et plus précisément une grille, avec les adjacences correspondantes. C'est donc par un rapide rappel des types de graphes les plus usuels que nous débuterons cette section avant de présenter une famille de graphes ayant une propriété spécifique aux segmentations de graphes par ensemble de sommets ainsi qu'une famille de grilles appartenant à cette catégorie. Cependant, il sera ensuite mis en évidence que ces grilles ne résolvent pas tout et que, dans le cas 2D, elles sont en fait un substitut à la segmentation de grilles par ensemble d'arêtes.

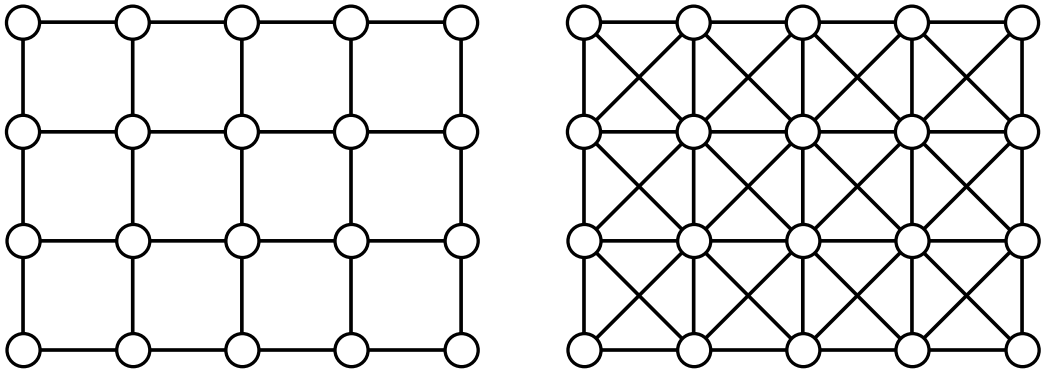
2.5.1 Grilles usuelles

Les graphes usuels pour la segmentation d'images sont les grilles équipées d'une des adjacences suivantes relatives à la dimension de la grille :

- la 4-adjacence ou la 8-adjacence pour une grille de dimension 2 ;
- la 6-adjacence, la 18-adjacence ou la 26-adjacence pour une grille de dimension 3 ;
- une des V_d^n-adjacences (avec $1 \leq d \leq n$) pour, dans le cas général, une grille de dimension n.

Les graphes de segmentation usuels sont donc les grilles $G = (S, A_d^n)$ où S est un sous-ensemble de $\mathbb{Z}^n$ du type $[O; d_1] \times [0; d_2] \times \ldots \times [0; d_n]$.

Des exemples de grilles usuelles en dimension 2 se trouvent dans la figure 2.8.



(a) (b)

FIGURE 2.8 – Grille de taille $(5, 4)$ en dimension 2 équipée de : (a) la 4-adjacence ; (b) la 8-adjacence.

2.5.2 Grilles de fusion parfaite

Dans [56], un autre type d'adjacence est décrit comme convenant à la segmentation de graphes et ayant une propriété de finesse des ensembles frontières de sommets.

Avant d'en arriver à la définition de ce type d'adjacence, nous allons tout d'abord présenter le cadre qui a motivé cette recherche et présenter la famille de graphes qui en résulte.

2.5.2.1 Fusion de régions

La fusion de régions dans un graphe [163, 176] peut s'avérer nécessaire lorsque le résultat d'une segmentation donne un résultat contenant de trop nombreuses régions, comme, par exemple, le résultat de certaines lignes de partage des eaux présentées dans le chapitre 3. On dit alors que le résultat est sur-segmenté.

Plutôt que de tenter de modifier la méthode de segmentation, des méthodes de fusion de régions furent également développées afin d'en réduire le nombre. Ainsi, on fusionne successivement des paires de régions voisines entre elles jusqu'à ce que l'on atteigne un certain critère d'arrêt, comme un nombre fixé de régions par exemple. Le choix des régions à segmenter ainsi que celui de la condition d'arrêt se font selon des critères propres à chaque application et qui ne seront donc pas abordées ici. La fusion de deux régions consiste simplement en la suppression des éléments de l'ensemble frontière qui sont adjacents aux deux régions concernées uniquement.

Dans certaines configurations, il arrive que la fusion de deux régions, pourtant toutes deux voisines des mêmes éléments de l'ensemble frontière, ne puisse se faire. En effet, si la suppression d'un élément commun de l'ensemble frontière adjacent à une troisième région est indispensable pour les fusionner, la suppression de cet élément entraînerait ainsi la fusion de trois régions en une seule. Or ceci n'est pas souhaité. Ce problème fut mis en évidence par Theo Pavlidis dans [163], section 5.6, *"When three regions meet"* ("Quand trois régions se rencontrent").

Exemple 2.27.
Sur la figure 2.6d, la région induite par le sommet i et celle induite par les sommets k et l ne peuvent fusionner car, en ôtant le sommet j de l'ensemble frontière, cela entraînerait la fusion simultanée de la région induite par les sommets a, b et f.

Il est à noter que ce problème ne peut se produire qu'avec des ensembles frontières de sommets puisque, comme déjà vu dans la section 2.3, une arête d'un ensemble frontière est adjacente à exactement deux composantes connexes de son graphe dual, empêchant ainsi toute fusion d'une troisième région par sa suppression. C'est pourquoi, seul le cas d'ensembles frontières de sommets est traité dans [56].

2.5.2.2 Graphes de fusion

La classe des **graphes de fusion** est l'ensemble des graphes pour lesquels, pour tout ensemble frontière de sommets, toute région peut être fusionnée avec une autre.

Cette classe de graphes semble limitée puisqu'elle ne permet pas nécessairement de fusionner des régions adjacentes à un même sommet de l'ensemble frontière. En effet, s'il y a plus de deux régions adjacentes à un même sommet de l'ensemble frontière, celui-ci ne peut être enlevé pour procéder à la fusion de deux d'entre elles puisque la troisième se retrouverait fusionnée par la même occasion. Toutefois, les graphes de fusion offrent l'intéressant théorème ci-dessous.

Théorème 2.28 (Adapté à partir de [56], théorème 33)**.**
Le graphe G est un graphe de fusion si et seulement si tout ensemble frontière de sommets sur G est mince.

On déduit aisément de la propriété 2.13 et du théorème 2.28 le théorème 2.29 ci-dessous.

Théorème 2.29.
Le graphe G est un graphe de fusion si et seulement si tout ensemble frontière de sommets sur G est fin.

Compte-tenu du défaut des graphes de fusion évoqué plus haut concernant les sommets de l'ensemble frontière adjacents à plus de deux régions, le théorème 2.29 nous fait prendre conscience que la condition de finesse évoquée dans la section 2.2.3 comme étant souvent souhaitée dans le cadre de segmentation de graphes ne s'avère pas suffisante lorsque l'on souhaite procéder à des fusions de régions.

Pour remédier à cela, une autre classe de graphes fut définie.

La classe des **graphes de fusion parfaite** est l'ensemble des graphes pour lesquels, pour tout ensemble frontière de sommets $K \subseteq S$, toute paire de régions adjacentes à un même sommet de K peut être fusionnée en retirant l'ensemble des sommets de l'ensemble frontière qui leur sont toutes deux adjacentes. Ainsi, tout sommet d'un ensemble frontière est strictement adjacent à deux régions.

Les graphes de fusion parfaite offre donc un cadre idéal à la fusion de graphe puisque l'opération de fusion est simplifiée, étant donné qu'il n'y a plus à vérifier si le sommet à retirer n'est pas adjacent à une troisième région.

Les graphes de fusion parfaite peuvent être caractérisés simplement comme le montre le théorème suivant.

Théorème 2.30 ([56], théorème 41)**.**
Le graphe G est un graphe de fusion parfaite si et seulement si aucun sous-ensemble de sommets de $S(G)$ n'induit le graphe présenté sur la figure 2.9.

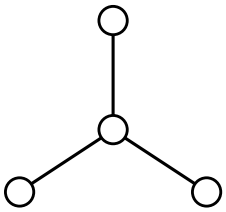


FIGURE 2.9 – Graphe minimal empêchant la fusion parfaite.

Les théorèmes 1.18 et 2.30 nous permettent ainsi de déduire le corollaire suivant.

Corollaire 2.31.
Tout graphe dérivé est un graphe de fusion parfaite.

Les relations entre les classes de graphes décrites ci-dessus sont telles que :
- tout graphe dérivé est un graphe de fusion parfaite ;
- tout graphe de fusion parfaite est un graphe de fusion.

Remarque 2.32.
Couplé au théorème 2.29, on déduit des inclusions ci-dessus que tout ensemble frontière de sommets dans un graphe dérivé est fin (et mince).
Ce résultat rejoint ainsi celui présenté dans la section précédente avec le théorème 2.26.

2.5.2.3 Grilles de fusion parfaite

Maintenant que nous avons présenté ce qu'était un graphe de fusion parfaite, nous allons aborder le cas des grilles de fusion parfaite qui offre ainsi une alternative aux grilles usuelles, présentées précédemment. En effet, le théorème 2.30 permet de se rendre compte que les grilles usuelles en dimension 2 et 3 ne sont pas des graphes de fusion parfaite. Une grille de fusion parfaite est donc une grille avec une adjacence particulière la dotant des propriétés de fusion et de finesse décrites ci-avant pour les ensembles frontières de sommets. En voici une définition simplifiée par rapport à celle présentée dans [56], section 7.

On appelle **maille de** $\mathbb{Z}^n$ un ensemble M de n-uplets $x = (x_1, \ldots, x_n)$ tel que $\forall x, y \in M$, $\sqrt{(x_1 - y_1) + (x_2 - y_2) + \ldots + (x_n - y_n)} \leq \sqrt{n}$.

Exemple 2.33.
Voici quelques exemples de mailles dans différentes dimensions :
- *dans* $\mathbb{Z}^1$ *(1D) :* $\{0, 1\}$ *et* $\{4, 5\}$;
- *dans* $\mathbb{Z}^2$ *(2D) :* $\{(0,0), (0,1), (1,0), (1,1)\}$ *et* $\{(4,4), (4,5), (5,4), (5,5)\}$;
- *dans* $\mathbb{Z}^3$ *(3D) :* $\{(0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1)\}$.

Des exemples illustrés se trouvent dans la figure 2.10.

On appelle **maillage de fusion parfaite dans** $\mathbb{Z}^n$ un ensemble de mailles de $\mathbb{Z}^n$ tel que :
- l'intersection entre deux mailles de cet ensemble est d'au plus un élément de $\mathbb{Z}^n$, et
- chaque élément de $\mathbb{Z}^n$ est inclus dans exactement deux mailles de cet ensemble.

Remarque 2.34.
Il est à noter qu'il existe 2^{n-1} *maillages de fusion parfaite distincts dans* $\mathbb{Z}^n$.
Ceux-ci sont identiques à une translation d'un maillage près.

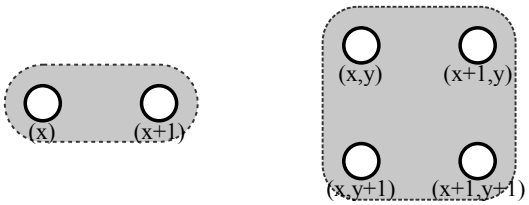


FIGURE 2.10 – (a) Maille 1D ; (b) Maille 2D.

Des exemples de maillages de fusion parfaite en dimension 1 et 2 se trouvent dans la figure 2.11.

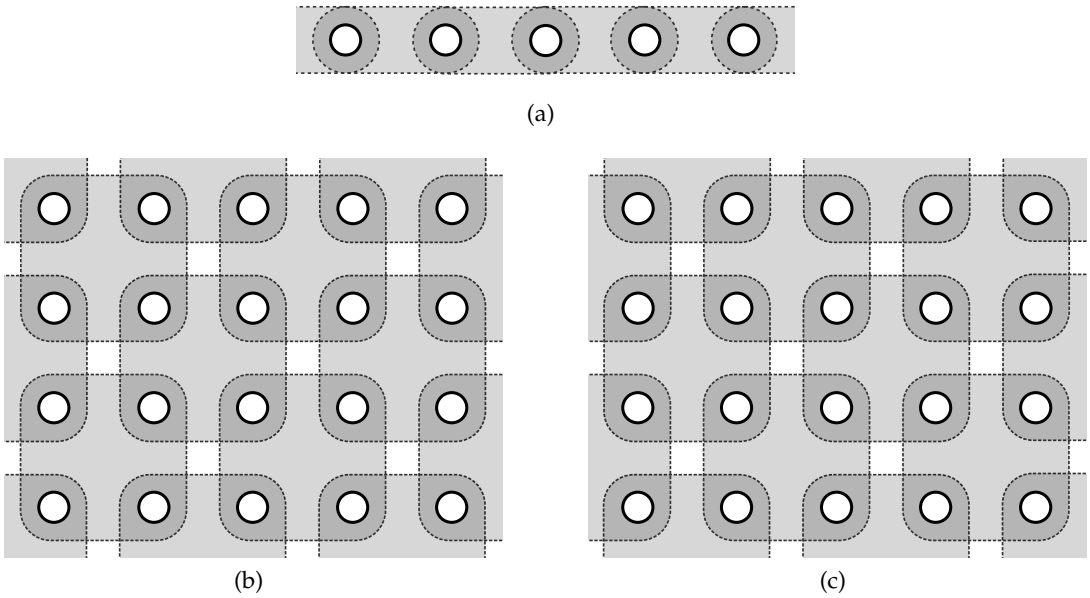


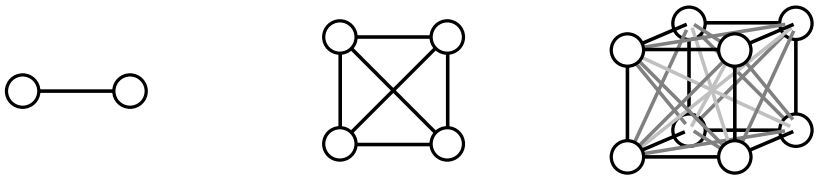
FIGURE 2.11 – (a) Le maillage de fusion parfaite 1D ; (b,c) Les deux maillages de fusion parfaite 2D.

On appelle **grille de fusion parfaite** une grille $G = (S, A)$ de dimension n telle que, pour un maillage de fusion parfaite $\mathcal{M}$ dans $\mathbb{Z}^n$, tout couple de sommets de S appartenant à une même maille de $\mathcal{M}$ sont reliés par une arête.

En conséquence de cette définition, toute maille de $\mathcal{M}$ est une clique maximale de G.

Remarque 2.35.
Puisque, comme mentionné dans la remarque 2.34, il existe 2^{n-1} *maillages de fusion parfaite distincts dans* $\mathbb{Z}^n$*, il existe donc* 2^{n-1} *grilles de fusion parfaite distinctes de dimension n ([56], propriété 56).*

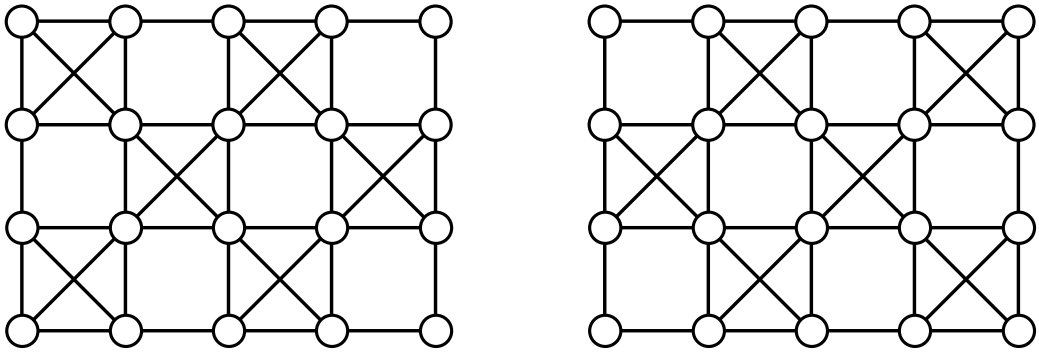
Des exemples de cliques maximales sur des mailles se trouvent dans la figure 2.12. Dans le cas de la représentation 3D (figure 2.12c), les différents niveaux de gris représentent les arêtes de différentes "longueurs" afin d'aider à la visualisation (arêtes plus claires pour les arêtes plus "longues"). Cette convention sera surtout utile à la bonne compréhension des figures 2.14 et 2.18 qui sont beaucoup plus chargées.



(a) (b) (c)

FIGURE 2.12 – Graphes formant une clique maximale sur une maille dans le cas : (a) 1D ; (b) 2D ; (c) 3D.

Des exemples de grilles de fusion parfaite en dimension 2 se trouvent dans la figure 2.13 et en dimension 3 dans la figure 2.14.



(a) (b)

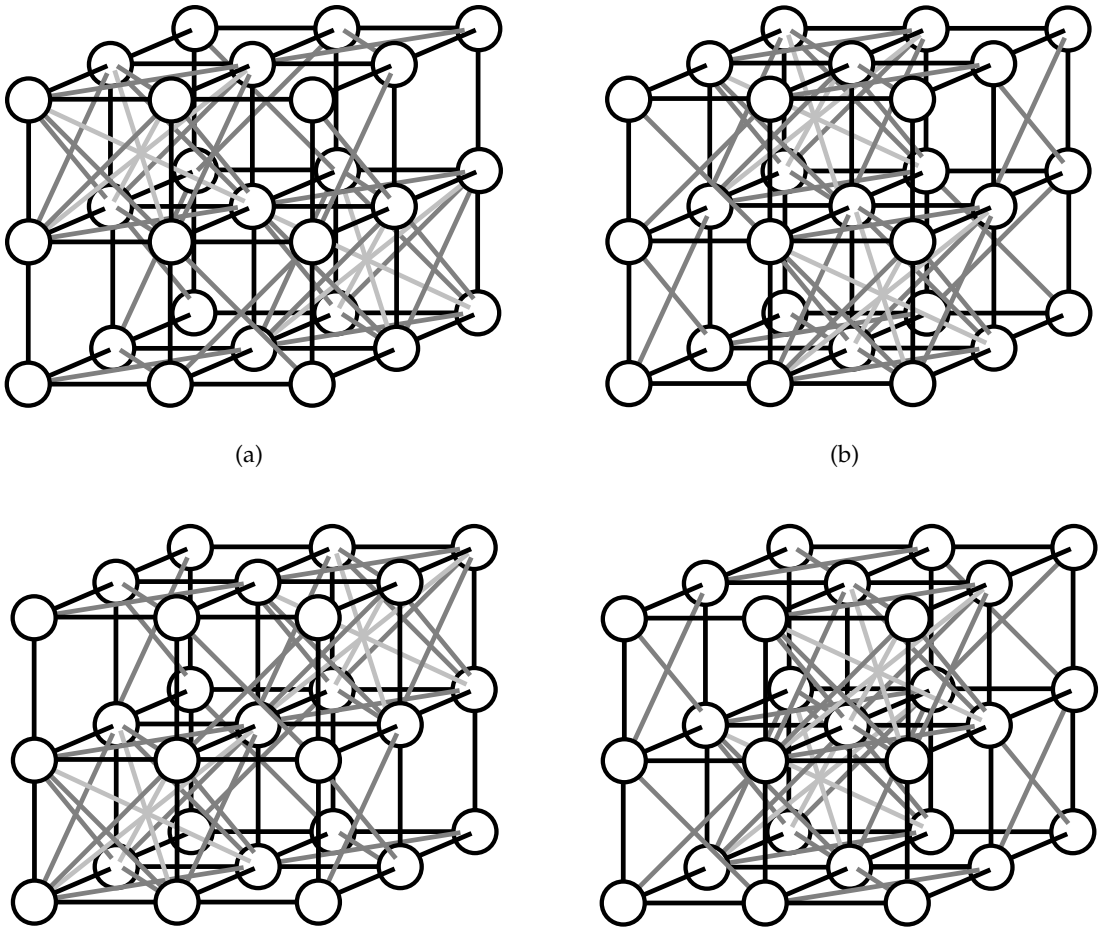
FIGURE 2.13 – Les deux grilles de fusion parfaite de dimension 2 avec $S = [0;4] \times [0;3]$.

Les grilles de fusion parfaite sont des grilles qui peuvent être considérées comme "intermédiaires" entre les grilles usuelles, offrant ainsi une alternative garantissant la finesse d'un ensemble frontière de sommets ainsi que les propriétés adéquates à la fusion de régions.

Il existe néanmoins des inconvénients à l'utilisation des grilles de fusion parfaite dus au fait qu'elles ne soient pas invariantes par translation. Ainsi, le résultat obtenu sur une grille de fusion parfaite est déterminé par le choix de celle-ci puisqu'elle définit les relations d'adjacence entre les sommets du graphe.

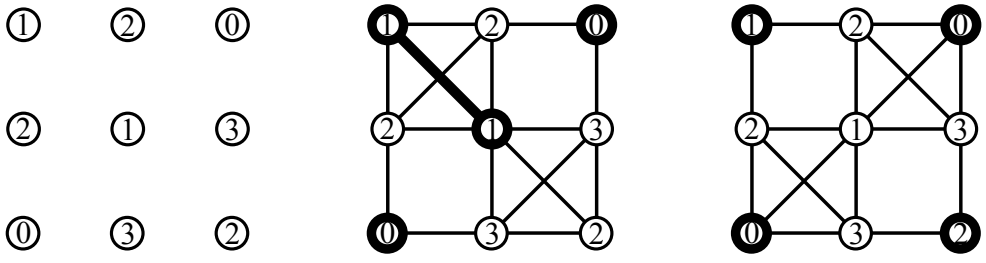
Exemple 2.36.
Voici deux exemples en 2D où les deux grilles donnent des résultats différents :

- *la recherche des minima régionaux d'une application de poids sur les sommets (voir figure 2.15) ;*
- *la fusion de régions à travers un ensemble frontière de sommets (voir figure 2.16 : les régions A et D ont pu être fusionnées sur la figure 2.16c alors que non sur la figure 2.16d et, réciproquement, ce sont les régions B et C qui ont pu l'être sur la figure 2.16d alors que cela est impossible sur la figure 2.16c).*



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

FIGURE 2.14 – Les quatre grilles de fusion parfaite de dimension 3 avec $S = [0;3]^3$.



(a) (b) (c)

FIGURE 2.15 – (a) Application de poids P sur $[0;2]^2$; (b,c) $Min(P)$ pour les deux grilles de fusion parfaite 2D sur $[0;2]^2$.

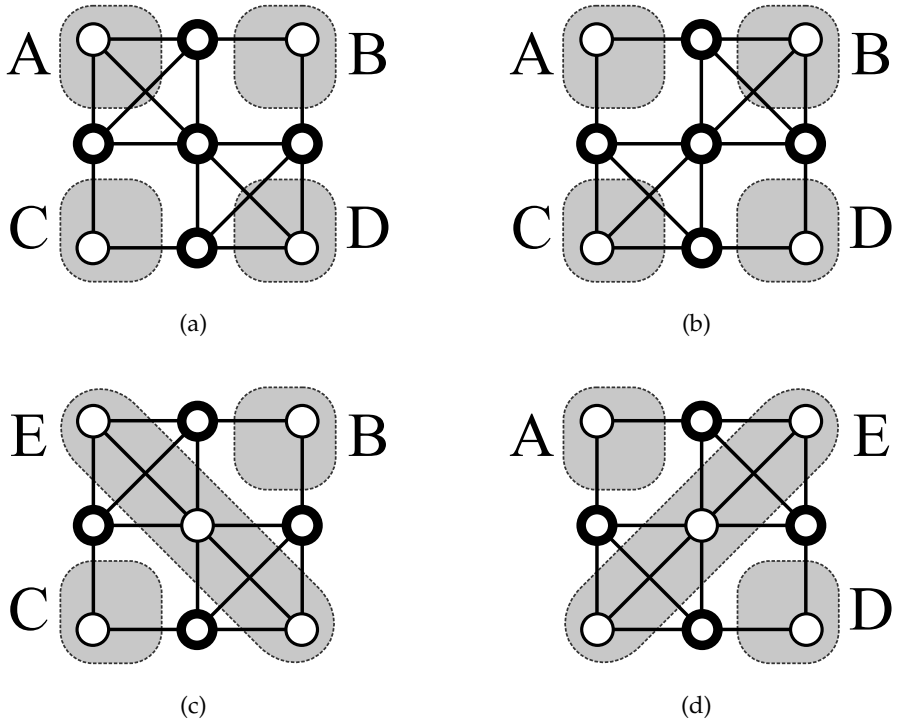


FIGURE 2.16 – (a,b) Ensemble frontière de sommets K (en gras) sur les deux grilles de fusion parfaite 2D, respectivement G_1 et G_2. (c) K (en gras) après fusion des régions A et C sur G_1 ; (c) K (en gras) après fusion des régions B et D sur G_2.

2.5.3 Grilles de fusion parfaite et graphes dérivés

Par définition d'une grille de fusion parfaite, on déduit aisément de la propriété 1.16 que toute grille de fusion parfaite, quelle que soit sa dimension n, est un graphe dérivé ([56], propriété 54).

Il est alors possible de retrouver le graphe dont la grille de fusion parfaite est le graphe dérivé. Dans le cas d'une grille 2D de taille infinie, ce graphe n'est en fait rien d'autre qu'une grille 2D équipée de la 4-adjacence, comme le montre la figure 2.17.

Il est cependant à noter qu'une telle correspondance n'existe pas en dimension 3 comme le montre la figure 2.18.

Ceci nous permet de nous rendre compte que les propriétés énoncées ci-avant concernant les ensembles frontières de sommets dans une grille de fusion parfaite de dimension 2 ne sont rien d'autres que les propriétés des ensembles frontières d'arêtes dans une grille de dimension 2 équipée de la 4-adjacence héritées à travers la dérivation puisque celle-ci conserve les propriétés d'adjacence (voir la remarque 1.21).

Comme on peut le voir dans la figure 2.17, en dimension 2, une grille de fusion parfaite finie ne correspond pas exactement à la dérivée d'une grille équipée de la 4-adjacence. En effet, on observe que son graphe d'origine possède en plus une arête pour chaque maille formant une clique qui se trouve dans "angle" de la grille.

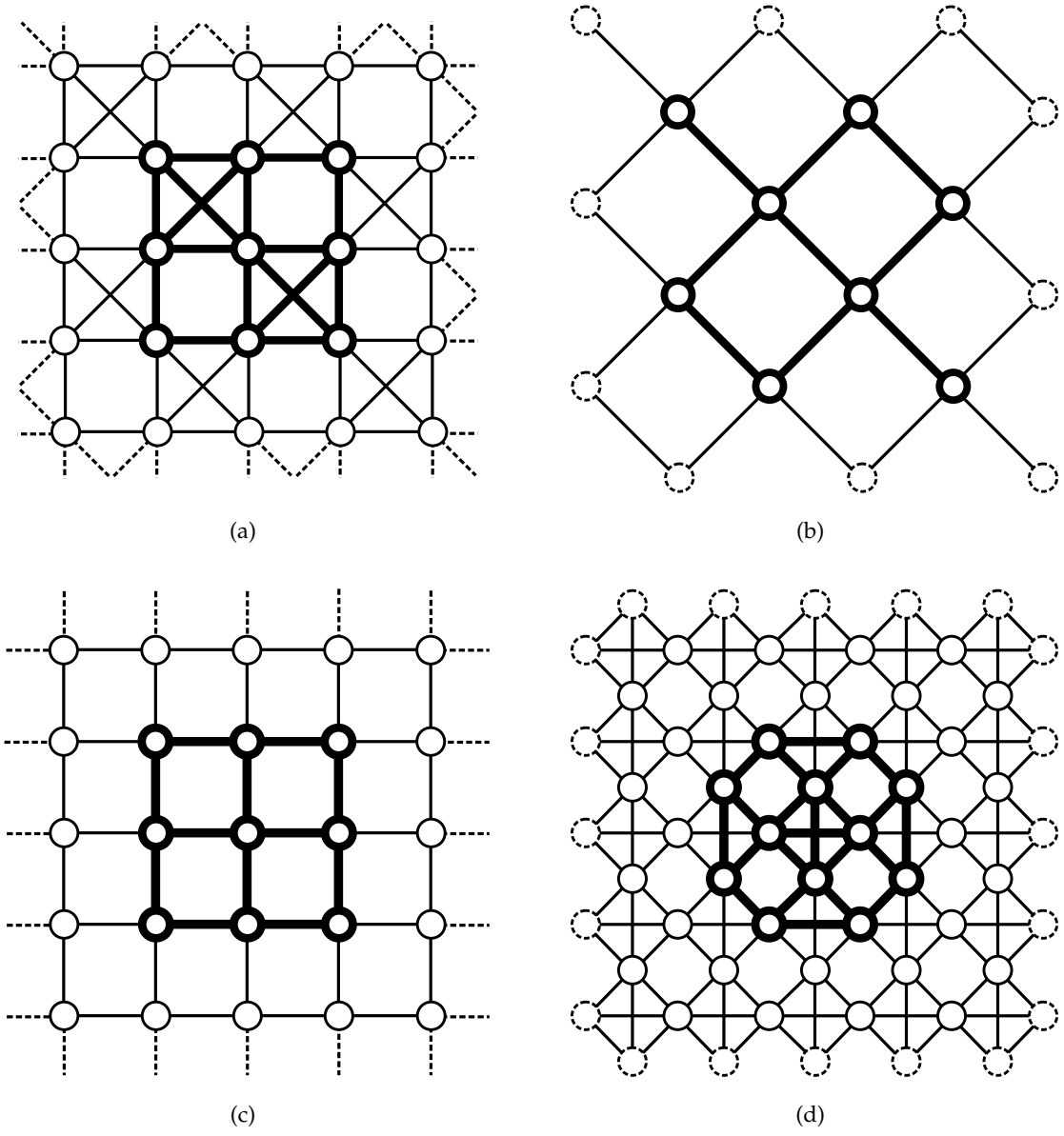
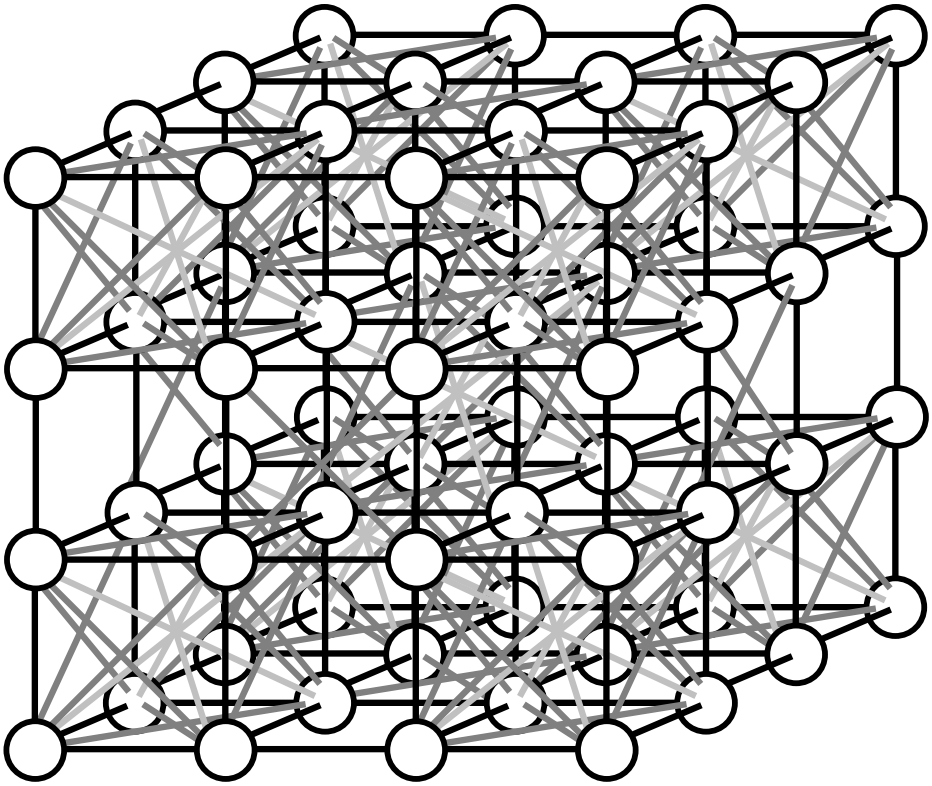
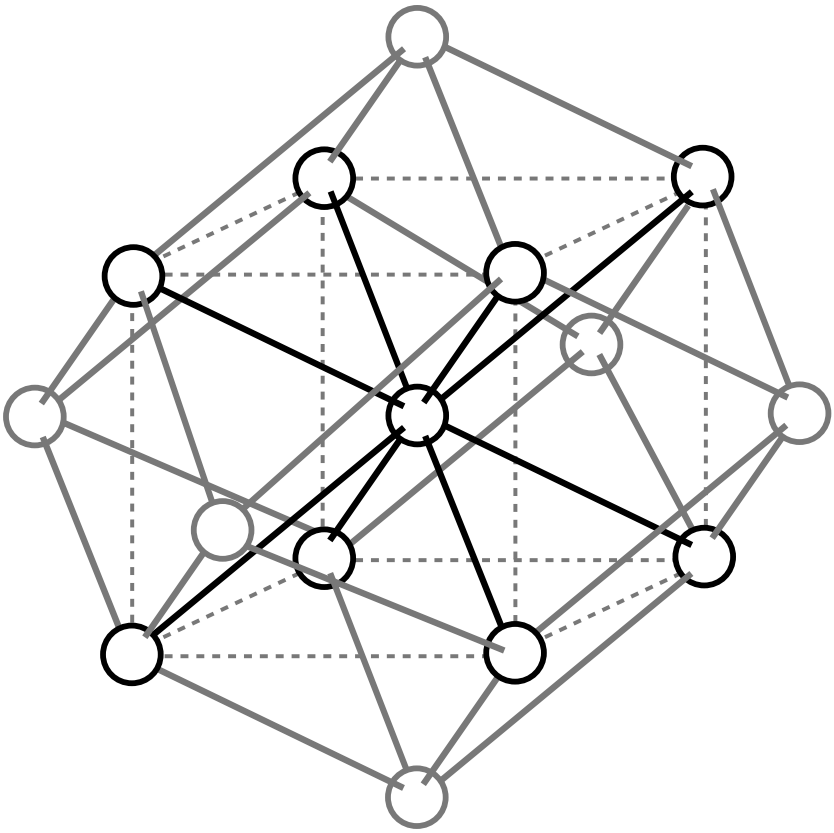


FIGURE 2.17 – En dimension 2 : (a) grille de fusion parfaite G_1 de taille infinie ; (b) graphe G_1' dont le dérivé est G_1 ; (c) grille de taille infinie équipée de la 4-adjacence G_2 ; (d) graphe G_2' dérivé de G_2. En gras se trouvent un sous-graphe de G_1, respectivement G_2, et son graphe correspondant dans G_1', respectivement G_2'. On note que $G_1 = G_2'$ et $G_2 = G_1'$ (à une rotation de 45° et un facteur d'échelle près sur cette représentation).



(a)



(b)

FIGURE 2.18 – (a) Grille de fusion parfaite G en dimension 3 de taille $(3,3,3)$. (b) Graphe G' dont le dérivé est G ; les traits en pointillés ne sont là que pour aider à visualiser le graphe dans l'espace en 3D en mettant en évidence le cube dans lequel s'inscrit les adjacences du sommet, représentées en noir, dont le dérivé donne la maille 3d formant une clique au "centre" de la figure (a).

2.6 Quel type de segmentation choisir ?

En conclusion de ce chapitre nous faisons un bilan des diverses possibilités qui ont été présentées concernant la segmentation de graphes afin de déterminer celle qui réunit les propriétés adéquates à la segmentation et que nous utiliserons par la suite.

Un ensemble frontière de sommets sépare des sommets par l'intermédiaire d'autres sommets. De manière analogue, un ensemble frontière d'arêtes sépare des arêtes par l'intermédiaire d'autres arêtes. Une coupe se distingue de ces deux notions puisqu'elle sépare des sommets par l'intermédiaire d'arêtes. Le tableau 2.1 récapitule les propriétés inhérentes à chacun de ces types de segmentation.

	Partition de sommets	Frontière fine	Fusion parfaite de régions	Sommet isolé comme région
Ensemble frontière de sommets dans un graphe quelconque	-	-	-	OK
Ensemble frontière de sommets dans un graphe de fusion parfaite	-	OK	OK	OK
Ensemble frontière d'arêtes	OK	OK	OK	-
Coupe	OK	OK	OK	OK

Table 2.1 – Propriétés des différents types de segmentation de graphe.

Le tableau 2.1 nous permet donc de déduire que la notion de coupe permet de réunir le plus grand nombre de propriétés utiles dans le cadre de la segmentation.

Le fait qu'un ensemble de sommets frontière ne soit pas toujours mince peut poser problème dans certaines applications. De plus, il a été mis en évidence, que dans des algorithmes de fusion de région [56, 176], la seule caractéristique de minceur d'un ensemble frontière n'est pas suffisante à éviter tout problème. Ainsi, un sommet de l'ensemble frontière qui est voisin de plus de deux composantes connexes du graphe dual pourra empêcher la fusion de deux régions de se faire correctement puisque, par sa suppression de l'ensemble frontière, il liera en même temps une autre composante connexe non désirée. Un ensemble frontière d'arêtes, de par le simple fait qu'une arête ne peut être adjacente à plus de deux composantes connexes du graphe dual, évitera ces soucis. Ceci est également valable pour les coupes qui, pour les mêmes raisons, ne posent aucune difficulté à la fusion de régions.

Les grilles de fusion parfaite semblent être un cadre propice à l'utilisation d'ensembles frontières de sommets puisqu'elles offrent les propriétés de finesse et de fusion parfaite. Toutefois, le fait qu'elles n'offrent pas d'invariance par translation peut poser problème dans certains cas. Dans le cas 2D, on se rend compte qu'une grille de fusion parfaite n'est rien de plus (à quelques arêtes supplémentaires en plus dans le cas fini) que la dérivée d'une grille équipée de la 4-adjacence. Les propriétés qu'on y trouve ne sont donc rien de plus que les propriétés des ensembles frontières d'arêtes qui sont conservées avec la dérivation.

Il peut paraître tentant de vouloir passer d'un ensemble frontière d'arêtes à un ensemble frontière de sommets et réciproquement selon l'algorithme dont on a besoin et le type de résultat désiré.

Toutefois, la recherche d'un ensemble frontière d'arêtes à partir d'un ensemble frontière de sommets nécessite certains traitements particuliers en raison, entre autres, des sommets intérieurs puisqu'un ensemble frontière de sommets n'est pas nécessairement mince.

L'opération inverse, la recherche d'un ensemble frontière de sommets à partir d'un ensemble frontière d'arêtes, n'en est pas plus simple. En effet, on pourrait imaginer récupérer, pour chaque paire de régions, les sommets extrémités des arêtes formant l'ensemble frontière qui se trouvent dans l'une ou l'autre des deux régions. Cependant, dans certains cas, une telle opération ne garantit pas d'obtenir un nombre de région moindre ou même ne pas obtenir un ensemble frontière de sommets comme le montre la figure 2.19.

En conclusion, outre le fait qu'il n'est pas toujours possible de se restreindre à travailler dans une certaine catégorie de graphes, les résultats présentés dans ce chapitre nous permettent donc de préconiser l'utilisation des coupes comme type de segmentation de graphes. C'est dans cette optique que s'orientera la suite de ce mémoire.

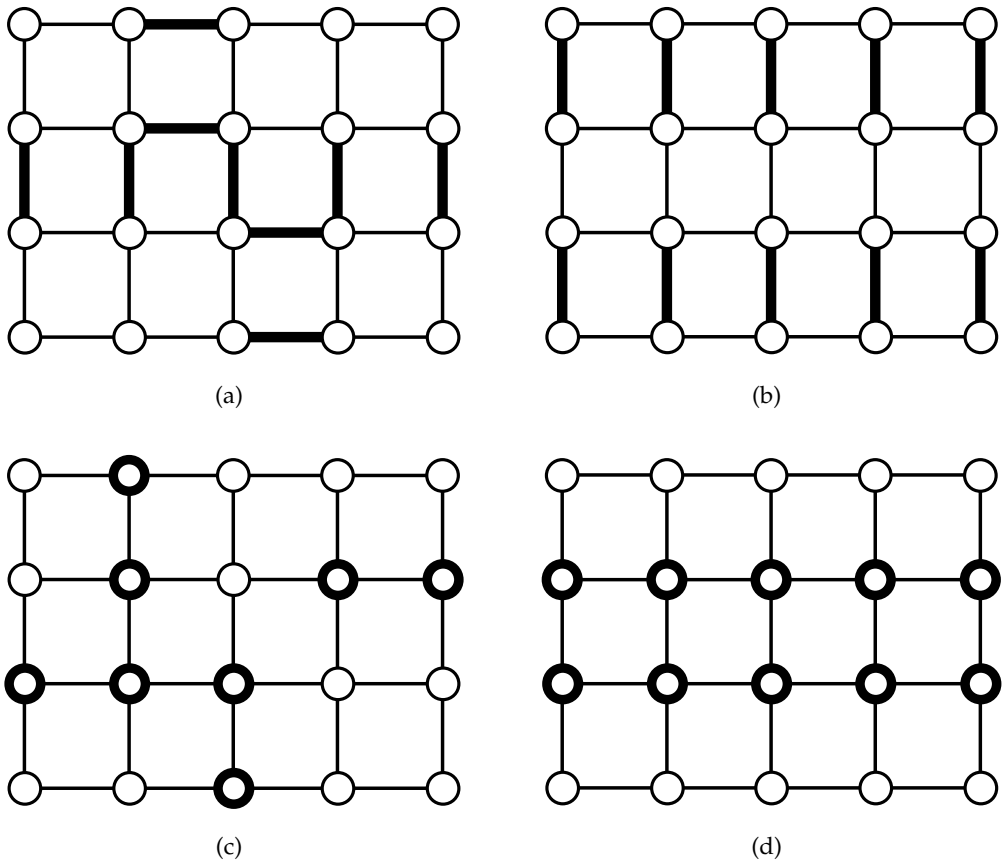


FIGURE 2.19 – Graphe G avec (en gras) : (a,b) ensembles frontières d'arêtes, respectivement A_1 et A_2 ; (c,d) ensembles de sommets, respectivement S_1 et S_2, obtenus en prenant les sommets d'un des deux régions adjacente à chaque arête de A_1 et A_2 respectivement. L'ensemble S_1 est un ensemble frontière, mais pas S_2 dont le nombre de régions ne correspond même pas à celui de A_2.

Ligne de partage des eaux

3

La ligne de partage des eaux est un concept emprunté au registre de la topographie et utilisé dans le traitement d'images depuis presque trente ans. Nous entamons ce chapitre en décrivant l'idée de base des lignes de partage des eaux afin de mettre en évidence comment ce concept peut être utile dans le traitement d'images et dans la segmentation de graphes. Les travaux concernant les lignes de partage des eaux ainsi que les algorithmes de recherche sont trop nombreux pour faire un passage en revue exhaustif. Nous détaillons donc uniquement quelques définitions de ligne de partage des eaux parmi les plus usitées ou les plus récentes, que ce soit sur graphe non-orienté à sommets ou à arêtes valués, afin d'en faire une comparaison en montrant certains liens existant entre ces différents paradigmes.

En topographie, un **bassin versant**, aussi appelé **bassin hydrographique** ou **bassin d'attraction**, est un territoire où toutes les eaux ruissellent vers son point le plus bas, qui constitue ainsi un minimum régional. Cet écoulement se fait en suivant la pente la plus forte. Les eaux se rejoignent alors pour former une rivière ou un fleuve (par exemple le bassin versant du Rhône ou de l'Allier), un lac (par exemple le bassin versant du Lac d'Annecy) ou une nappe souterraine avant d'atteindre finalement une mer ou un océan.

La **ligne de partage des eaux (LPE)** désigne la limite géographique naturelle entre deux bassins versants. C'est l'ensemble des points à partir desquels de l'eau est susceptible de s'écouler dans des directions différentes amenant chacune à un minimum régional différent. Il est à noter que les lignes formées par ces points ne correspondent pas obligatoirement aux lignes de crêtes (lignes qui relient l'ensemble des points culminants du relief).

La figure 3.1 illustre les termes définis ci-dessus tandis que la figure 3.2 illustre la notion de LPE dans le cadre topographique de l'Europe.

Avant d'aller plus loin, il est à préciser que, dans le but de mettre en évidence les liens existant entre les définitions de LPE présentées dans ce chapitre, certaines d'entre elles ont été généralisées afin de pouvoir être transposées du cadre des graphes non-orientés à sommets valués à celui des graphes non-orientés à arêtes valuées.

De même, certaines définitions de LPE ne définissent pas une segmentation (au sens où nous l'avons présenté dans le chapitre 2) mais un ensemble de régions (en l'occurrence les bassins versants) ou une application de poids. Dans ces cas, toujours dans le but de pouvoir comparer les différents paradigmes, nous nous ramenons à une définition d'ensemble séparant ou d'ensemble frontière en exposant comment celui-ci découle de la définition d'origine.

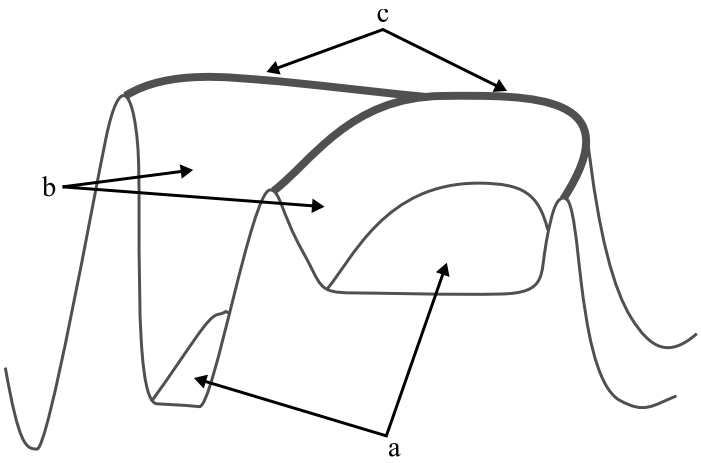


FIGURE 3.1 – Un relief topographique avec : (a) des minima régionaux ; (b) des bassins versants ; (c) LPE.

FIGURE 3.2 – Bassins versants et principales lignes de partage des eaux européennes (les zones blanches correspondent aux régions où l'eau s'écoule jusqu'a la mer ou l'océan sans rejoindre un des principaux fleuves indiqués sur la carte).

Sommaire du chapitre

Figures du chapitre

3.1 Ligne de partage des eaux, images et graphes

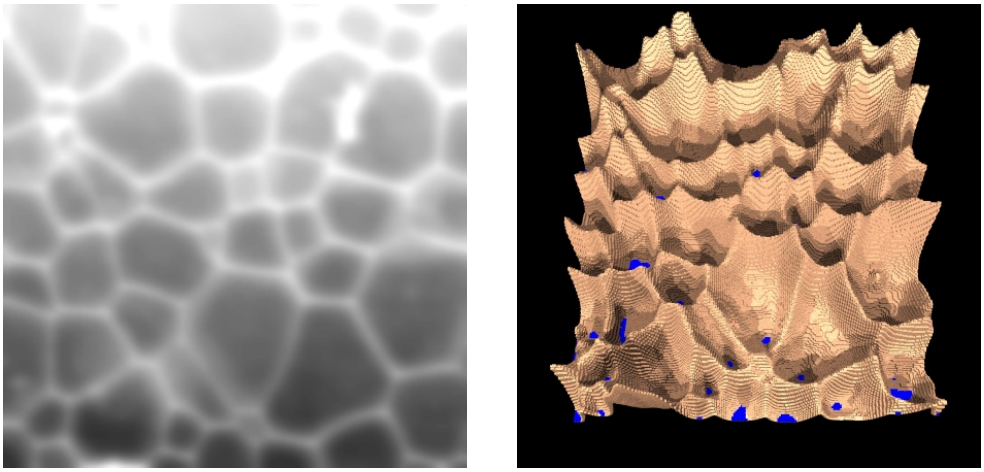
La notion de LPE apparaît dès le milieu du XIX^{ème} siècle [44, 111, 143, 177, 182], mais ce n'est que vers la fin du XX^{ème} siècle, en 1979, qu'elle fut utilisée pour la première fois dans le cadre du traitement d'images par H. Digabel et Christian Lantuéjoul [71]. Elle fut, dès lors, l'objet de nombreuses études pour finalement devenir un outil majeur de la morphologie mathématique pour la segmentation d'images avec, par exemple, les travaux de Serge Beucher [29, 30, 31], Fernand Meyer [144, 146, 148], Luc Vincent et Pierre Soille [206] qui fournirent des algorithmes efficaces pour les déterminer.

Nous allons tout d'abord rappeler la définition d'une image pour mettre en évidence la façon dont une LPE peut être appliquée dessus, puis nous verrons que son application a pu être généralisée à tout graphe non-orienté. C'est dans ce cadre que prendront place les définitions de LPE présentées dans ce chapitre.

3.1.1 Images, topographie et ligne de partage des eaux

Une image 2D en niveaux de gris, telle que présentée dans la section 1.5, peut être considérée comme un relief topographique : le niveau de gris de chaque pixel représente son altitude (voir figure 3.3). Un pixel noir représente le niveau le plus bas (niveau 0) et un pixel blanc le niveau le plus haut (niveau 255).

C'est sur le relief ainsi constitué que l'on peut procéder à la recherche de LPE de façon similaire au domaine de la topographie. Le but de cette recherche est, la plupart du temps, destiné à déterminer les frontières des régions qui composent l'image et ainsi de récupérer les contours d'un objet par exemple.



(a) (b)

Figure 3.3 – (a) Une image en niveau de gris ; (b) "Relief topographique" représenté en 3D associé à l'image (a).

Il est à noter que le domaine d'application de l'image n'est pas nécessairement celui des images en niveaux de gris, mais qu'il doit être un espace métrique de sorte à pouvoir calculer une distance (une "altitude") par rapport à un point référence pour chaque élément de l'image.

Nous verrons dans la sous-section 3.1.2 que la recherche d'une LPE ne se limite pas aux domaines de définition d'une image.

3.1.2 Graphes, images et LPE

Il a été vu dans la section 1.5 qu'une image associée à un voisinage donné était un graphe particulier. Dans la suite de ce chapitre, nous considérerons que toute image sera associée au 4-voisinage et qu'en conséquence elle sera équivalente à une grille à sommets valués équipée de la 4-adjacence. Dans un but de généralisation, nous emploierons donc de préférence une terminologie liée aux graphes (nous parlerons de sommets adjacents plutôt que de pixels voisins ou de poids d'un sommet plutôt que de valeur d'un pixel par exemple).

Il est à noter que c'est par le biais de cette interprétation que l'analogie entre pondération et relief a pu être transmise au domaine des graphes. C'est de cela que viennent certains termes de vocabulaire liés aux graphes tels que *"altitude"* ou *"plateau"* par exemple. Ainsi, l'altitude de connexion entre deux régions représente l'altitude du col le plus bas permettant de se rendre d'une région à l'autre.

En toute logique, si l'on se base sur la définition de la LPE topographique, la recherche d'une LPE sur un graphe non-orienté valué revient à chercher les sous-graphes représentant les bassins versants. Une fois les bassins versants déterminés, on peut alors en déduire l'ensemble séparant relatif aux minima régionaux qui en résulte et qui forme la LPE. Les LPE et les bassins versants du domaine des graphes doivent avoir des propriétés équivalentes (ou censées l'être) aux LPE et aux bassins versants du domaine de la topographie. Ainsi, on devine aisément que l'on doit être capable d'y retrouver les propriétés exposées ci-dessous de manière informelle :

P1. Une LPE est un ensemble frontière ou une coupe relative aux minima régionaux, en conséquence de quoi les bassins versants d'une LPE sont des extensions des minima régionaux.

P2. Pour tout bassin versant, il existe, dans le bassin, un chemin descendant allant de tout élément de la LPE qui lui est adjacent jusqu'au minimum régional que contient ce bassin.

P3. Pour tout élément d'un bassin versant, il existe un chemin descendant allant de cet élément jusqu'au minimum régional que contient ce bassin.

P4. Pour toute paire de bassins versants, l'altitude de connexion est la même que l'altitude de connexion des minima régionaux qu'ils contiennent respectivement.

P5. Une LPE est fine.

Il est à préciser que le terme "chemin" employé dans les propriétés P2 et P3 ne fait pas référence à la notion homonyme existant dans les graphes mais sert uniquement à se représenter aisément ces propriétés dans le cadre topographique.

Les LPE que nous présentons ici étant sur des graphes non-orientés, ces "chemins" mentionnés ici seront en fait interprétés en termes de chaînes dans la suite de ce chapitre.

Nous verrons par la suite que plusieurs tentatives de formalisations différentes de LPE ont été faites dans le but d'obtenir des algorithmes efficaces. Certaines de ces définitions trouvent leur place initiale dans tout graphe non-orienté à sommets valués (voir section 3.2) alors que d'autres, plus récentes, la trouvent dans tout graphe non-orienté à arêtes valuées (voir section 3.3).

Il sera montré que toutes ces définitions ne vérifient pas toujours les propriétés P1 à P5 énoncées ci-dessus (auxquelles nous ferons référence tout au long de ce chapitre).

Les notions abordées dans ce chapitre seront toutes considérées dans le cas de graphes non-orientés uniquement. C'est pourquoi, par la suite, nous considérerons le graphe non-orienté $G = (S, A)$.

Les définitions de LPE proposées dans les sections 3.2 et 3.3 dépendent des minima régionaux de la pondération du graphe, que celle-ci soit sur les sommets ou les arêtes. Nous considérerons dans ces sections que les applications de poids utilisées comportent au moins deux minima régionaux.

3.2 LPE sur graphes à sommets valués

Comme on l'a vu dans la section 3.1.1, le cadre initial de la LPE provient du domaine de la topographie et fut adapté au cadre des images puisque ces dernières peuvent être assimilées à un relief.

Nous verrons dans cette section différentes définitions de LPE initialement définies pour des graphes non-orientés à sommets valués et que nous avons pu généraliser au cas de graphes non-orientés à arêtes valuées dans un but de comparaison. C'est pourquoi, bien que nous soyons dans la section des graphes à sommets valués, des exemples sur des graphes à arêtes valuées seront également présentées pour illustrer ces définitions.

Il est cependant à préciser que les cinq propriétés souhaitées pour une LPE présentées ci-avant ne seront évaluées, dans cette section, que dans le cadre des graphes à sommets valués, qui est le cadre d'origine de ces définitions.

3.2.1 LPE par immersion

La **LPE par immersion**, aussi appelée **LPE par inondation**, est la méthode par laquelle la LPE fut introduite en tant qu'outil de morphologie mathématique en 1979 [30] (voir également [144, 206]).

Il est très facile de se représenter le "phénomène physique" engendré par cette méthode :

1. Imaginons une maquette représentant le relief topographique associé à une image.
2. On perce la maquette de part en part à chaque fond de vallée (chacun de ses minima régionaux).
3. On immerge progressivement la maquette dans un liquide, on observe alors de l'eau surgir des minima pour recouvrir partiellement la maquette et former des lacs.
4. Au fur et à mesure de l'immersion de la maquette, on érige des digues à chaque endroit où les liquides en provenance de minima différents se rencontrent au-dessus de la maquette.

Les digues ainsi construites constituent la LPE par immersion de l'image associée.

Une représentation en coupe de ce processus se trouve dans la figure 3.4.

Remarque 3.1.
Il est à noter que l'édification des digues est, en soi, sans importance. On pourrait très bien se contenter de marquer sur la maquette les endroits où de l'eau en provenance de différents minima se rencontre et ses marques formeraient alors la LPE par immersion, mais l'image de la construction de digues est couramment employée.

Cette méthode semble s'éloigner du principe d'écoulement des eaux dans différents bassins versants puisqu'il est ici question de faire monter le niveau de l'eau. Pourtant, il

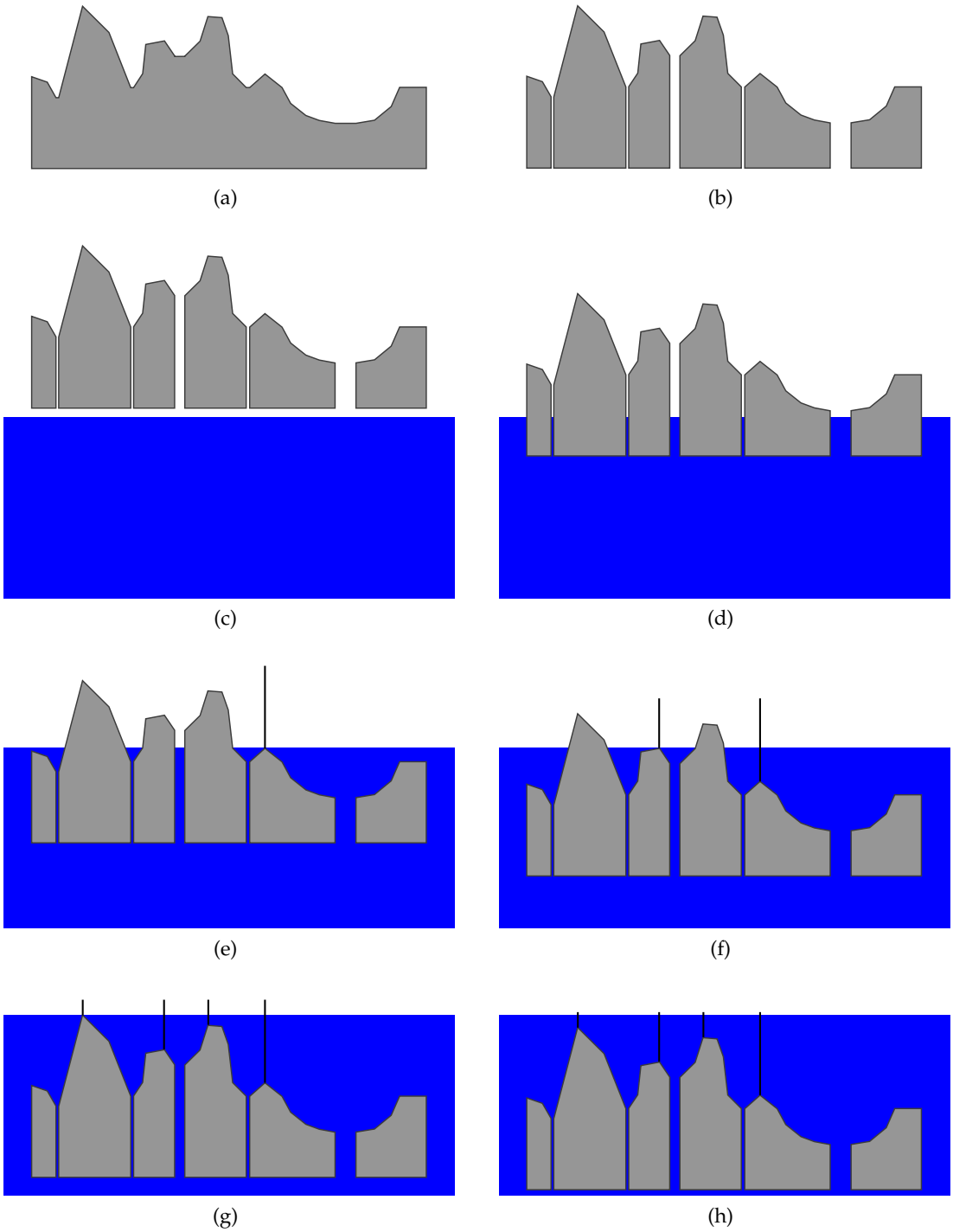


FIGURE 3.4 – (a) Relief topographique associé à une image 1D ; (b) Relief topographique (a) "percé" aux minima régionaux ; (c) à (h) Immersion progressive de (b) et construction de "digues" pour éviter les mélanges de liquide provenant de minima/sources différent(e)s.

s'agit ici de trouver une séparation, notre fameuse "ligne" (de partage des eaux), délimitant ainsi les régions entourant chaque minimum régional du relief considéré.

Ce procédé n'est pas sans rappeler la notion d'amincissement présentée en section 2.2. En effet, plus l'immersion avance, plus l'ensemble encore "émergé" diminue. Néanmoins, cet amincissement doit toujours commencer par les éléments de moindre altitude. C'est pourquoi Fernand Meyer définit l'amincissement par immersion dans [144] pour les graphes non-orientés à sommets valuées et que nous avons généralisé ci-dessous pour la rendre applicable aux graphes à arêtes valuées.

Définition 3.2 (Amincissement par immersion)**.**
Soit une application de poids P sur les sommets ou les arêtes. Soit E l'ensemble d'éléments de G sur lequel s'effectue la pondération.
On dit qu'un ensemble $E' \subseteq E$ est un ***amincissement par immersion*** *(de G pour P) si :*
- $\mathcal{D}(E') = Min(P)$, *ou*
- *il existe un amincissement par immersion E' de G pour P et un élément $e \in E'$ tels que :*
  - $E' = E'' \setminus \{e\}$,
  - E' *est un amincissement de* E'', *et*
  - $P(e)$ *est minimal.*

Des illustrations de cette définition se trouvent dans les figures 3.5 et 3.6.

Remarque 3.3.
Par définition, si l'ensemble de sommets ou d'arêtes E est un amincissement par immersion de G pour P, alors $\mathcal{D}(E)$ est une extension de $Min(P)$.

Un amincissement par immersion K correspondant à un ensemble d'élément du même type que celui sur lequel se fait la pondération peut être obtenu par suppression itérative des éléments de moindre altitude qui sont adjacents à une unique composante connexe de $\mathcal{D}(K)$.

La définition d'amincissement par immersion nous permet ainsi de définir la LPE par immersion ci-dessous.

Définition 3.4 (LPE par immersion)**.**
On dit que L est une ***LPE par immersion*** *(de G pour P) si L est un ensemble frontière qui est un amincissement par immersion.*

Par cette définition, on cherche donc à déterminer un ensemble frontière en partant des minima régionaux et en les étendant pour les faire devenir des bassins versants.

Remarque 3.5.
Toute LPE par immersion est un ensemble frontière.
De plus, dans un graphe valué, il peut exister plusieurs LPE par immersion distinctes.

Soient L une LPE par immersion de G pour P et M un minimum régional de G pour P.

La composante connexe de $\mathcal{D}(L)$ qui contient M est le **bassin versant (par immersion)**, ou **bassin d'attraction (par immersion)**, relatif à M (de G pour P). Il est à noter que, par définition, les bassins versants forment une extension des minima régionaux (qui est couvrante dans le cas d'un ensemble frontière d'arêtes).

La définition 3.4 débouche tout naturellement sur l'algorithme proposé par Fernand Meyer dans [144] qui revient à étendre le plus possible les minima régionaux avec une

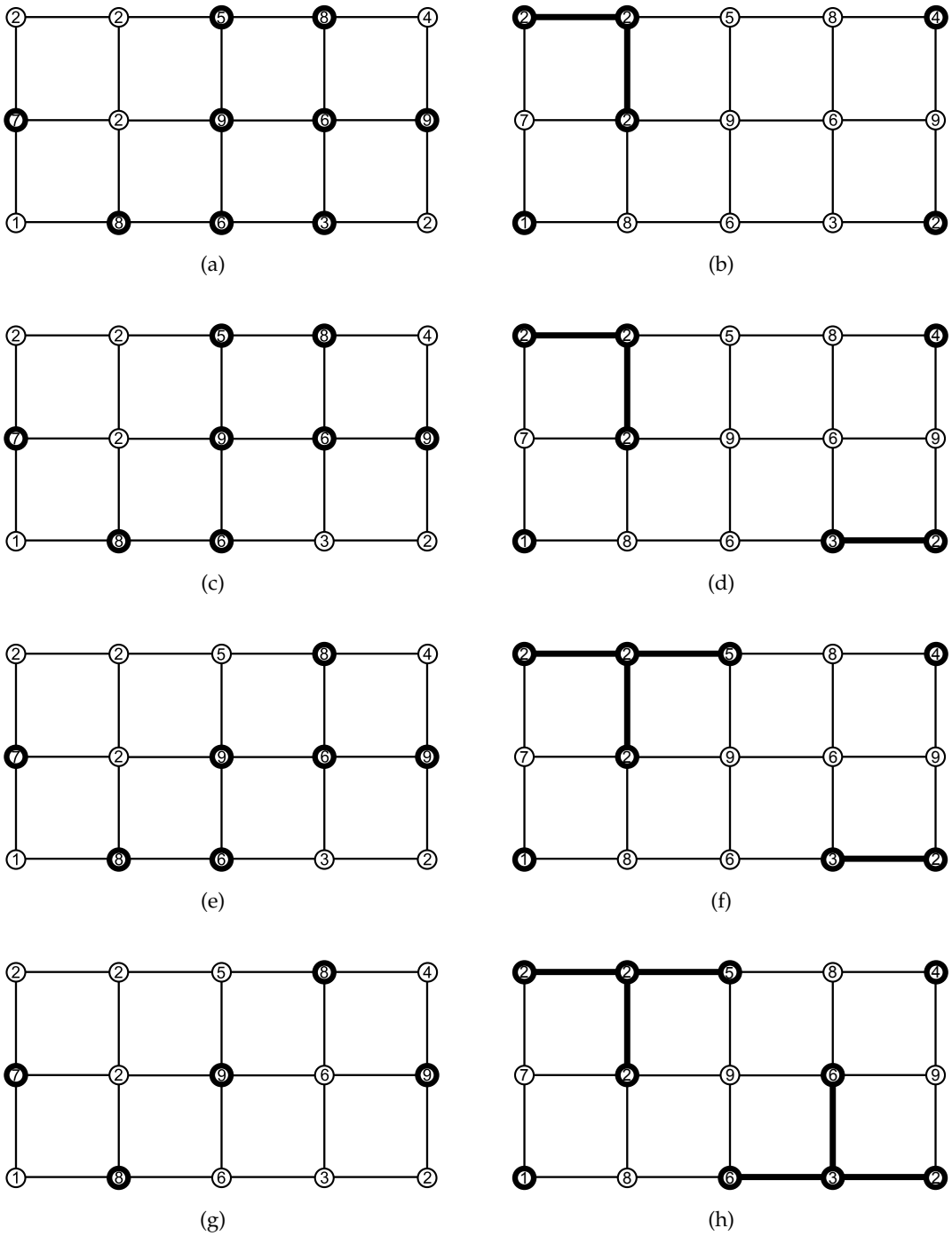


FIGURE 3.5 – Graphe G et application de poids P sur les sommets avec (en gras) : (a,c,e,g) quatre amincissements de sommets par immersion de sommets dans G pour P ; (b,d,f,h) leurs graphes duaux respectifs. Le sous-graphe en gras dans (b) est $Min(P)$ et l'ensemble de sommets en gras dans (g) est une LPE par immersion.

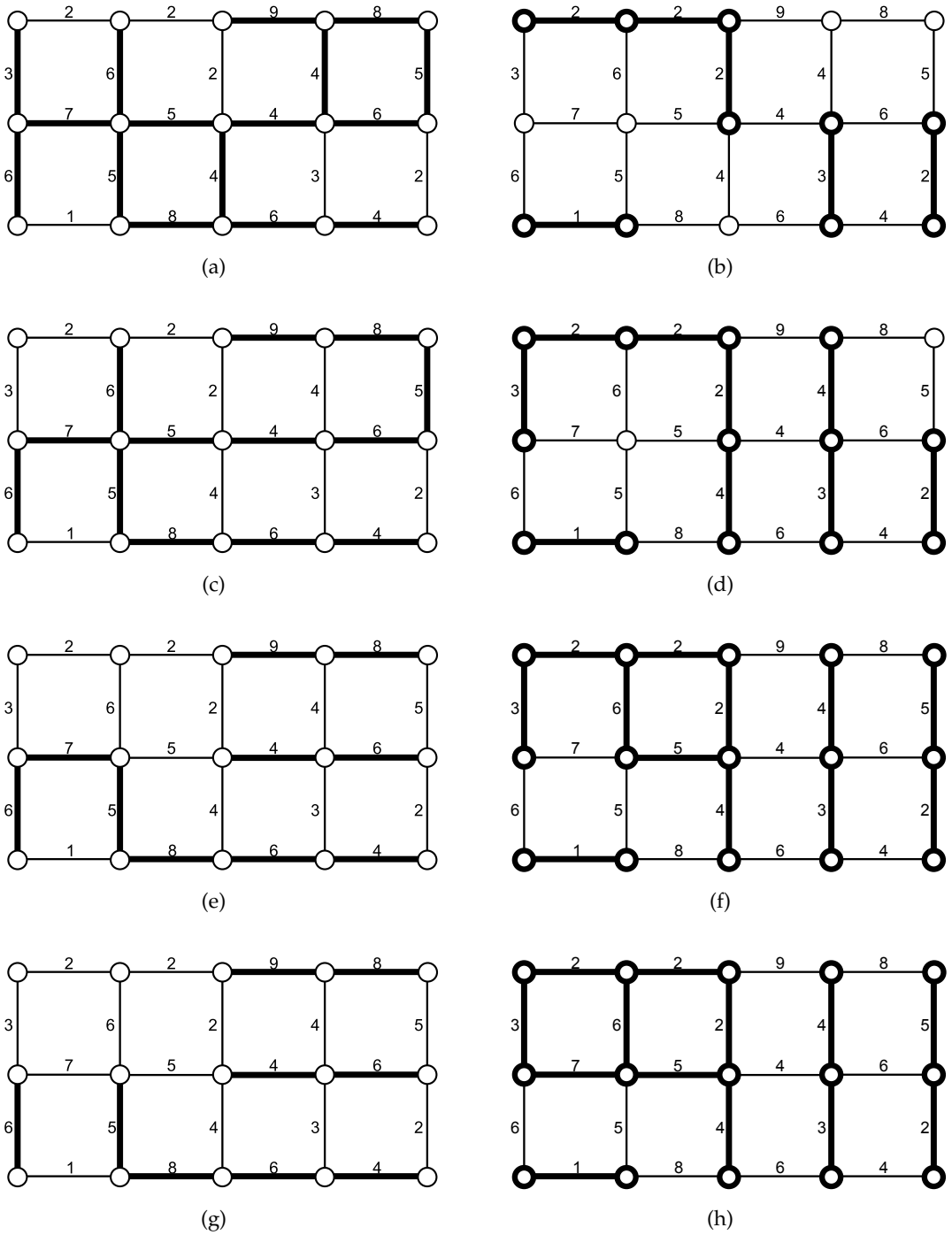


FIGURE 3.6 – Graphe G et application de poids P sur les arêtes avec (en gras) : (a,c,e,g) quatre amincissements d'arêtes par immersion d'arêtes dans G pour P ; (b,d,f,h) leurs graphes duaux respectifs. Le sous-graphe en gras dans (b) est $Min(P)$ et l'ensemble d'arêtes en gras dans (g) est une LPE par immersion.

priorité d'extension dépendant du poids des sommets comme nous le détaillons ci-dessous (pour le cas originel d'une LPE par immersion sur graphe à sommets valués).

Une LPE par immersion sur graphe à sommets valués peut être obtenue avec l'algorithme suivant :

1. Prendre l'ensemble séparant de sommets $L \subseteq S$ tel que $\mathcal{D}(L) = Min(P)$.
2. Enlever de L son sommet adjacent à une seule composante connexe de $\mathcal{D}(L)$ qui est de plus faible poids.
3. Répéter l'étape précédente jusqu'à idempotence.

L'ensemble frontière de sommets L résultant est une LPE de sommets par immersion.

Bien que cet algorithme soit le plus populaire, il en existe d'autres basés sur des principes similaires [174, 206].

Un exemple de LPE de sommets par immersion sur une image 2D en niveaux de gris, accompagnée des représentations topographiques correspondantes aux étapes intermédiaires, se trouve dans la figure 3.7.

Remarque 3.6.
Une LPE par immersion sur graphe à sommets valués ne satisfait que la propriété P1.

Un illustration des remarques 3.5 et 3.6 se trouve dans la figure 3.8.

En effet, pour un graphe et une application de poids identiques, il y a deux LPE possibles (figures 3.8b et 3.8c). De plus, la LPE de la figure 3.8c n'est pas mince puisque le sommet de poids 7 est intérieur. On note également que, dans la figure 3.8b il n'y a pas de chaîne descendante allant du sommet de poids 7 au sommet de poids 1 qui est le minimum régional du bassin versant auquel il appartient. Toujours dans la figure 3.8b, il n'y a pas non plus de chaîne descendante du sommet de poids 4 de la LPE au sommet de poids 1 qui est le minimum régional d'un bassin versant auquel il est adjacent. Enfin, l'altitude de connexion entre les bassins versants est inférieure à celle entre leur minimum régional respectif.

3.2.2 LPE inter-sommets par immersion

La **LPE inter-sommets par immersion** fut introduite par Fernand Meyer dans [144] (voir également [31]). Elle repose sur le même "phénomène physique" que le LPE par immersion présentée dans la section 3.2.1 et se base également sur les graphes à sommets valués, mais, à la différence de cette dernière, ce n'était pas un ensemble frontière qui était recherché, mais une partition des sommets attribuant chacun à un bassin versant.

Il est à noter que cette formulation revient à chercher un ensemble frontière d'arêtes avec un algorithme très proche de celui de la LPE par immersion (voir section 3.2.1) comme montré ci-dessous.

Une LPE inter-sommets par immersion peut être obtenue avec l'algorithme suivant :

1. Prendre l'ensemble séparant d'arêtes $E \subseteq A$ tel que $\mathcal{D}(E) = Min(P)$.
2. Prendre le sommet de $S(\mathcal{D}(E))$ de plus faible poids qui a des arêtes adjacentes à une unique composante connexe de $\mathcal{D}(E)$. Ôter ces arêtes de E.
3. Répéter l'étape précédente jusqu'à idempotence.

On a alors E qui est une LPE inter-sommets par immersion et $\mathcal{D}(E)$ forme une extension couvrante des minima régionaux de G.

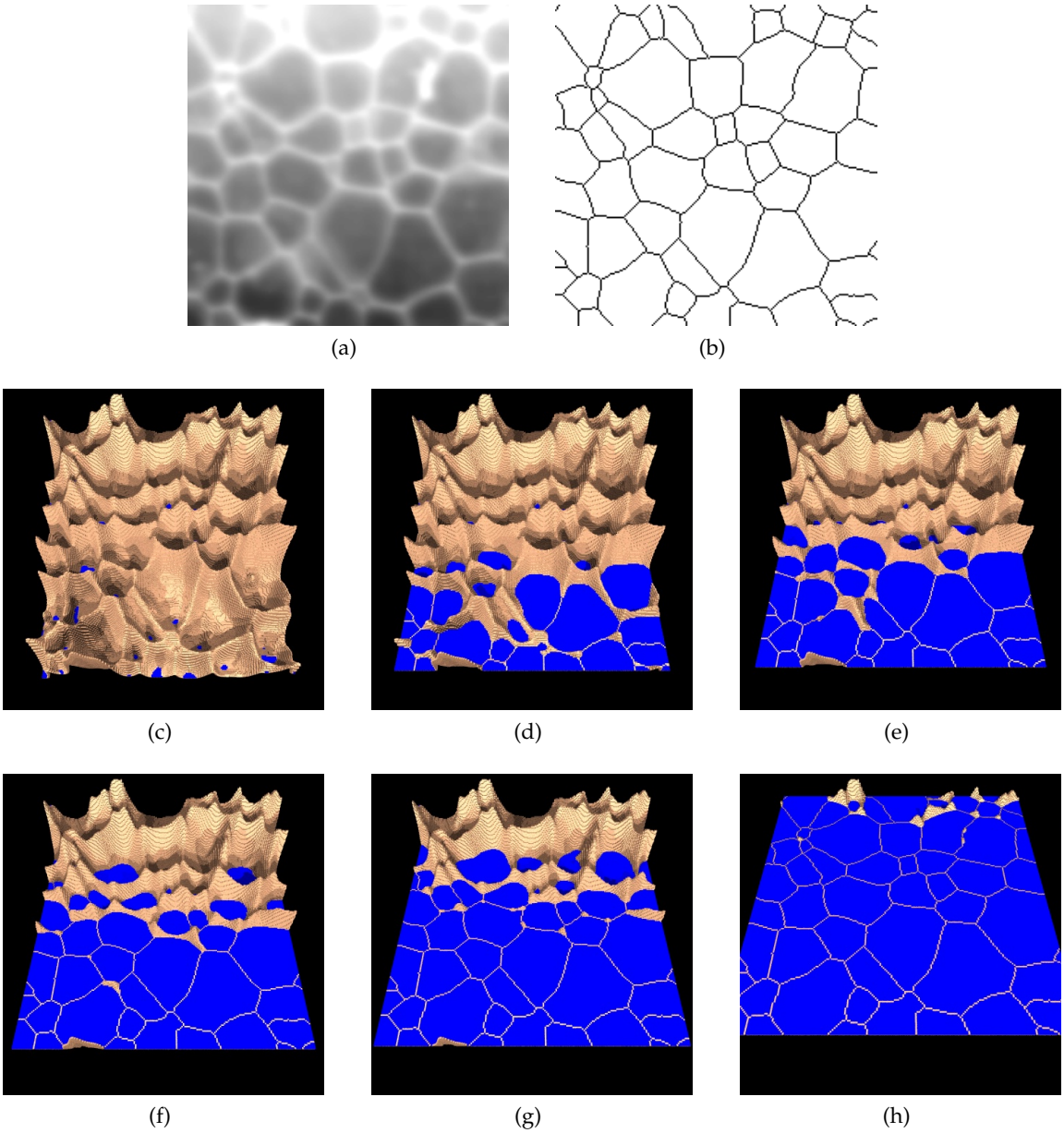


FIGURE 3.7 – (a) Une image ; (b) LPE par immersion de (a) ; (c) Représentation topographique de (a) avec en bleu les minima régionaux ; (d) à (h) Représentations topographiques d'étapes intermédiaires à l'immersion du "relief".

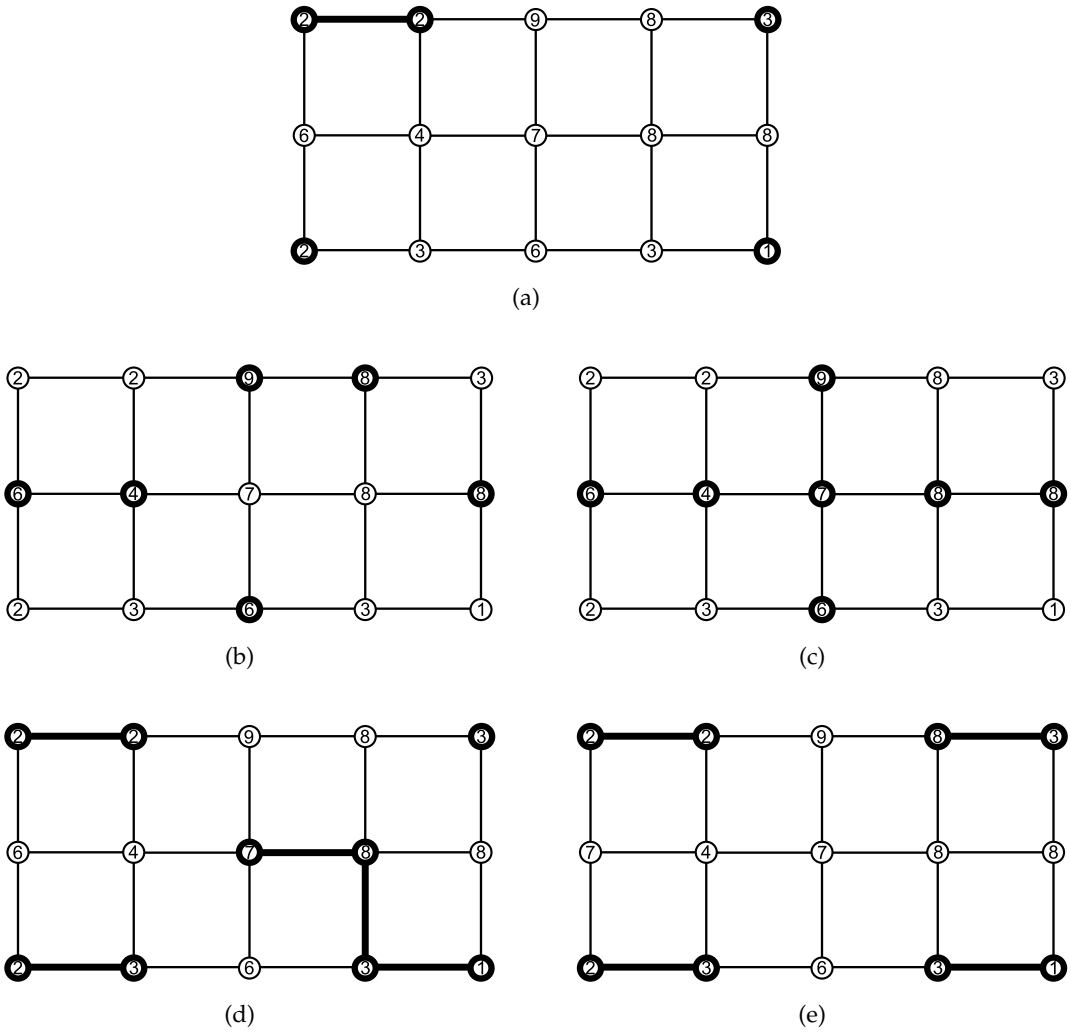


FIGURE 3.8 – Graphe G et application de poids P sur les sommets avec (en gras) : (a) $Min(G)$; (b,c) deux LPE de sommets par immersion de sommets ; (d,e) les bassins versants respectifs des LPEs des figures (b) et (c).

Remarque 3.7.
Toute LPE inter-sommets par immersion est un ensemble frontière d'arêtes.
De plus, dans un graphe valué, il peut exister plusieurs LPE par immersion distinctes.

Outre les avantages que procure l'utilisation d'ensembles frontières d'arêtes (voir chapitre 2), comme la propriété de finesse (théorème 2.15), à la différence de la LPE par immersion, la LPE inter-sommets par immersion permet d'avoir la propriété suivante.

Théorème 3.8.
Pour tout sommet $s \in S$, il existe une chaîne descendante de plus grande pente incluse dans le bassin versant contenant s et qui mène au minimum régional qu'il contient.

Remarque 3.9.
Les propriétés P1, P3 et P5 sont vérifiées par une LPE inter-sommets par immersion.
Les propriétés P2 et P4, quant à elles, ne sont pas applicables directement puisque la pondération originale se fait sur les sommets alors que les éléments composant la LPE sont des arêtes.

Une illustration des remarques 3.5 et 3.9 se trouve dans la figure 3.9.
En effet, pour un graphe et une application de poids identiques, il y a deux LPE possibles (figures 3.9b et 3.9c).

Il est à préciser qu'il existe une définition de la LPE inter-sommets par immersion reposant sur les forêts de chemins de moindre altitude (cette notion est définie dans la section 4.4 - [83]).

3.2.3 LPE par distance topographique

La notion de **LPE par distance topographique** est introduite en 1994 par Fernand Meyer [146] (voir également [154, 174]). Il n'est plus ici question de définir une LPE en tant qu'ensemble frontière, mais plutôt de déterminer les éléments qu'une LPE sépare : les bassins versants. Ceux-ci sont définis en se basant sur le phénomène originel du domaine de la topographie qu'est le ruissellement, autrement dit, l'écoulement des eaux le long d'une pente.
La définition de bassin versant topographique repose sur la distance topographique qui est une pseudo-métrique servant à faire le lien entre LPE et squelettes par zones d'influence (généralisation des diagrammes de Voronoï). Une formulation équivalente repose sur les définitions ci-après.

Définition 3.10 (Bassin versant topographique)**.**
Soit une application de poids P. Soit M un minimum régional de G.
*Le **bassin versant topographique** relatif à M (de G pour P) est le graphe induit par les sommets depuis lesquels M est le seul minimum régional atteignable par une chaîne de plus grande pente.*

La définition de bassin versant topographique n'est pas sans rappeler le théorème 3.8 concernant la LPE inter-sommets par immersion.

La définition originelle de LPE par distance topographique, pour un graphe G et une application de poids P est l'ensemble des bassins versants topographiques relatifs aux minima régionaux de G. Cependant, dans le cadre de notre comparaison, et afin prendre en considération un résultat se rapprochant d'une segmentation, nous la redéfinissons comme suit.

Définition 3.11 (LPE par distance topographique)**.**
Soit une application de poids P.
*La **LPE par distance topographique** est l'ensemble d'arêtes de G n'appartenant à aucun bassin versant topographique relatif à un minimum régional de G.*

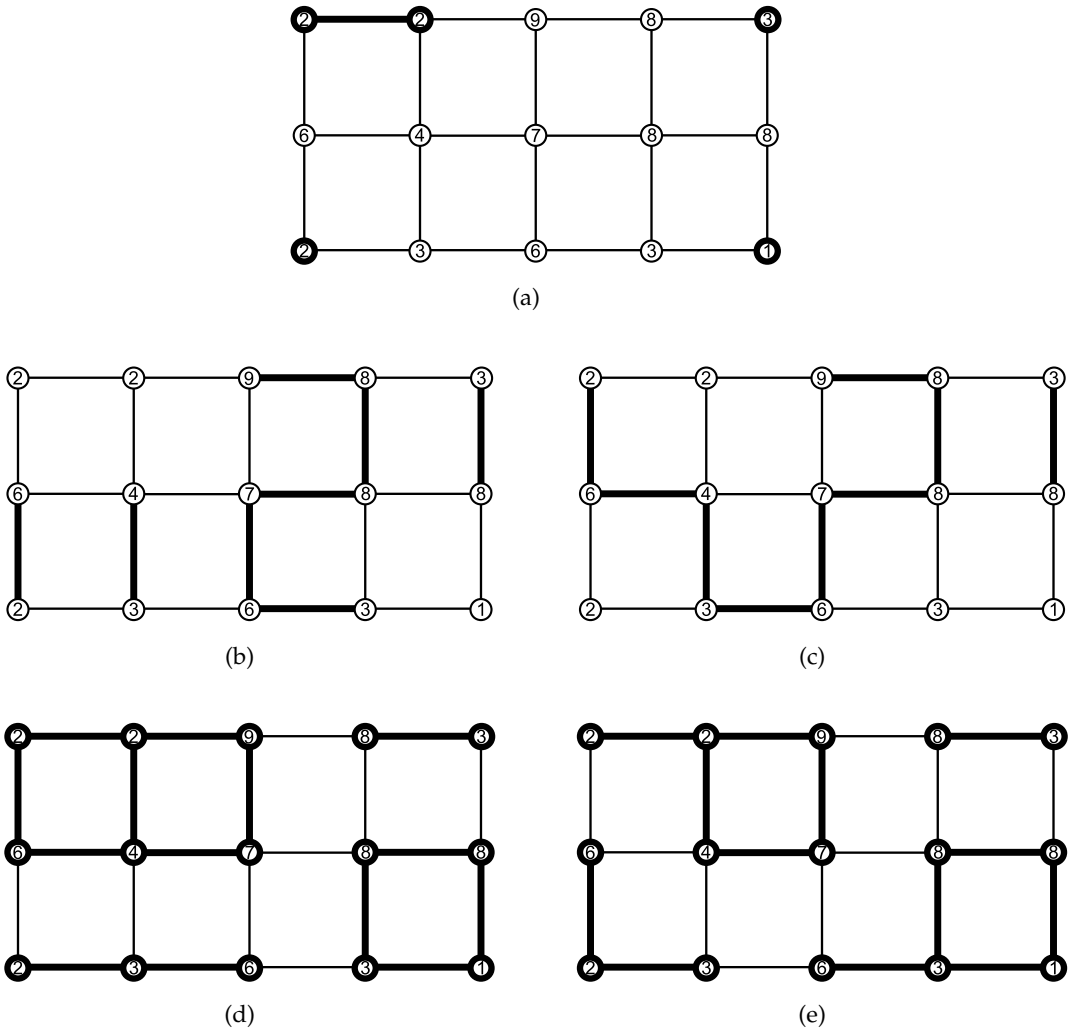


FIGURE 3.9 – Graphe G et application de poids P sur les sommets avec (en gras) : (a) Min(G) ; (b,c) deux LPE d'arêtes par immersion de sommets ; (d,e) les bassins versants respectifs des LPEs des figures (b) et (c).

Remarque 3.12.
Pour un graphe et une application de poids donnés, la LPE par distance topographique est un ensemble séparant d'arêtes unique qui n'est pas nécessairement un ensemble frontière.

Une illustration de la remarque 3.12 se trouve dans la figure 3.10.

En effet, dans la figure 3.10c, l'ensemble séparant d'arêtes n'est pas fin. De plus, là encore, il n'y a pas de chaîne descendante partant du sommet de poids 6 de la dernière rangée jusqu'au minimum régional en haut à gauche sur la figure 3.10a.

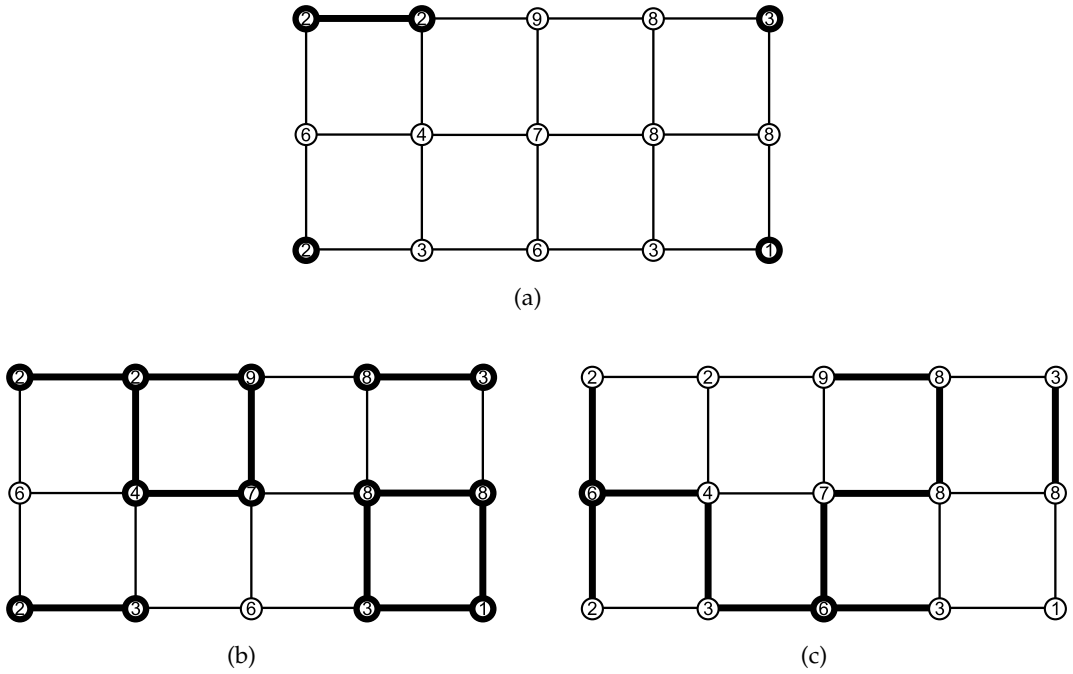


FIGURE 3.10 – Graphe G avec : (a) une application de poids P sur les sommets et (en gras) $Min(G)$; (b) la LPE topographique de G pour P et (en gras) ses différents bassins versants ; (c) les éléments de G qui n'appartiennent pas à sa LPE topographique.

Remarque 3.13.
D'après la remarque 3.12, les propriétés P1 et P5 ne sont donc pas vérifiées par une LPE par distance topographique.
D'après la définition de bassin versant topographique, la propriété P3 est vérifiée.
Enfin, les propriétés P2 et P4, quant à elles, ne sont pas applicables directement puisque la pondération originale se fait sur les sommets alors que les éléments composant la LPE sont des arêtes.

La différence principale entre LPE inter-sommets par immersion et LPE par distance topographique est que cette dernière, outre le fait d'être unique pour un graphe et une application de poids donnés, peut n'attribuer à aucun bassin versant certains sommets du graphe.

Cette différence est liée au théorème 3.14 ci-dessous.

Théorème 3.14.
Tout ensemble frontière d'arêtes obtenu par amincissement d'une LPE par distance topographique est une LPE inter-sommets par immersion.

Autrement dit, toute coupe induite par une extension couvrante relative à l'ensemble des bassins versants topographiques relatifs aux minima régionaux de G est une LPE inter-sommets par immersion.

3.2.4 LPE topologique

La **LPE topologique**, introduite en 1997 par Michel Couprie et Gilles Bertrand [54] avant d'être développée en 2005 [26, 55, 153], diffère radicalement des LPE précédentes puisqu'elle offre en résultat non pas un ensemble frontière, un ensemble séparant, une partition ou encore un ensemble de régions mais une application de poids. Il est cependant facile, à partir de cette application de poids, de définir un ensemble frontière. C'est cette définition que nous comparerons donc aux précédentes. De plus, ici encore, nous généraliserons la définition d'origine sur les graphes à sommets valués pour le cas des graphes à arêtes valuées.

Cette définition fut motivée par la volonté d'obtenir une LPE qui vérifie la propriété P4 : avoir la même altitude de connexion entre deux bassins versants qu'entre les minima régionaux qu'ils contiennent respectivement. Nous verrons plus loin que c'est en effet le cas, mais qu'aucune des quatre autres propriétés n'est vérifiée.

Là encore, il est possible de se représenter le "phénomène physique" engendré par cette méthode :

1. Imaginons une maquette représentant le relief topographique associé à une image.
2. On ravine petit à petit les zones qui ne forment pas un col entre deux minima pour en diminuer l'altitude (on considère qu'un ravinement ne peut pas créer de nouveaux minima) et ainsi ne conserver que des cloisons dont l'altitude est celle du col reliant les deux minima.

Le relief ainsi obtenu constitue la LPE topologique telle que décrite dans [26, 54, 55, 153]. Pour notre part, nous nous concentrerons sur les cloisons conservées qui constitueront notre (re)définition de LPE topologique.

Une représentation en coupe de ce processus se trouve dans la figure 3.11.

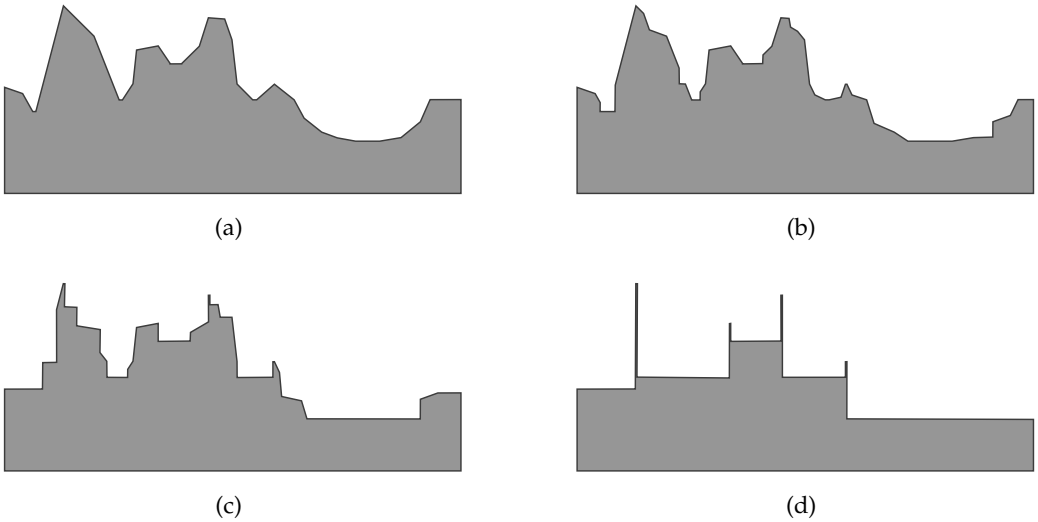


FIGURE 3.11 – (a) Relief topographique associé à une image 1D ; (b,c) Ravinements de (a) ; (d) Cloisonnement par ravinement de (a) qui forme une LPE topologique (dans sa définition d'origine) et dont les "pics" forment un ensemble frontière de (a) qui nous servira dans nos comparaisons et que nous appelons également, par abus de notation, LPE topologique de (a).

La définition ci-dessous décrit ce que l'on entend précisément par ravinement dans le cadre d'un graphe.

Définition 3.15 (Ravinement)**.**
Soit une application de poids P. Soit E l'ensemble d'éléments de G sur lequel s'effectue la pondération.
*On dit qu'une application de poids P' est un **ravinement** de P (pour G) si $\forall h \in \mathbb{R}$, G'^h_P est un amincissement de G^h_P.*

On appelle **élément destructible** un élément $e \in E$ de poids p et qui est adjacent à une unique composante connexe du graphe dual au seuillage inférieur à p (noté $\mathcal{D}(E^p_P)$).

Par définition, un élément destructible reste destructible si on l'abaisse tant que son poids ne descend pas en-dessous du poids le plus bas de ses éléments adjacents.

Un ravinement P' d'une application de poids P peut ainsi être obtenu par abaissement itératif d'éléments destructibles. L'opération peut ainsi être répétée sur un même élément jusqu'à ce qu'il ne soit plus destructible. Il est à noter que l'ordre de traitement des points destructibles est sans importance.

Propriété 3.16.
Par définition, le ravinement conserve les altitudes de connexion entre les minima régionaux.

Un ravinement ne peut pas créer de nouveaux minima régionaux ni en diminuer l'altitude, cela ne peut que les étendre.

Définition 3.17 (Cloisonnement par ravinement)**.**
Soit une application de poids P et P' un ravinement de P.
*S'il n'existe aucun ravinement distinct de P à l'application de poids P, alors on dira que P est un **cloisonnement par ravinement**.*
*De plus, si P' est un cloisonnement par ravinement, on dira que P' est un **cloisonnement par ravinement de** P.*

Remarque 3.18.
Il est à préciser que la définition de cloisonnement par ravinement faite ici correspond en fait à la définition originale de LPE topologique faite dans [26, 54, 55, 153].

Un cloisonnement par ravinement peut être calculé de façon quasi-linéaire en utilisant un arbre des composantes [152] de sorte à caractériser localement et en temps quasi-constant les points destructibles à abaisser [55].

Propriété 3.19.
Soit deux applications de poids P et P' sur les arêtes.
Si l'application de poids P' est un cloisonnement par ravinement, alors $Min(P')$ est une extension maximale de $Min(P)$.

Des illustrations de ces définitions se trouvent dans les figures 3.12 et 3.13.

La définition de cloisonnement par ravinement nous permet ainsi de définir la LPE topologique comme ci-dessous.

Définition 3.20 (LPE topologique)**.**
Soient une application de poids P sur les sommets et P' un cloisonnement par ravinement de P.
Soit E l'ensemble d'éléments de G sur lequel s'effectue la pondération.
*L'ensemble d'éléments de E qui n'appartiennent à aucun minimum régional de G pour P' forme une **LPE topologique** (de G pour P).*

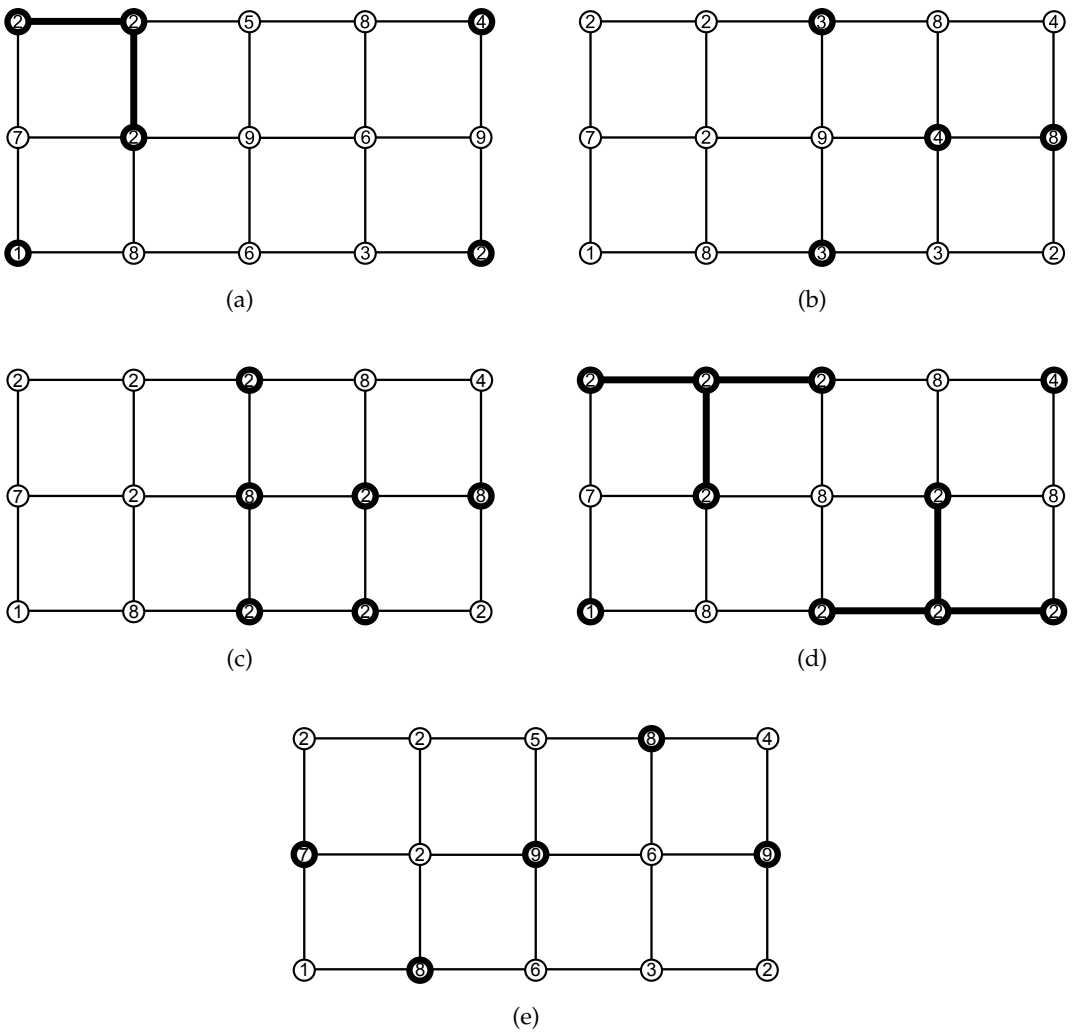


FIGURE 3.12 – Graphe G avec : (a) une application de poids P sur les sommets et (en gras) $Min(P)$; (b) un ravinement de P et (en gras) les sommets de poids différents ; (c) un cloisonnement par ravinement de P et (en gras) les sommets de poids différents ; (d) le cloisonnement par ravinement de P présenté en (c) et (en gras) ses minima régionaux ; (e) l'application de poids P et (en gras) une LPE topologique de G pour P.

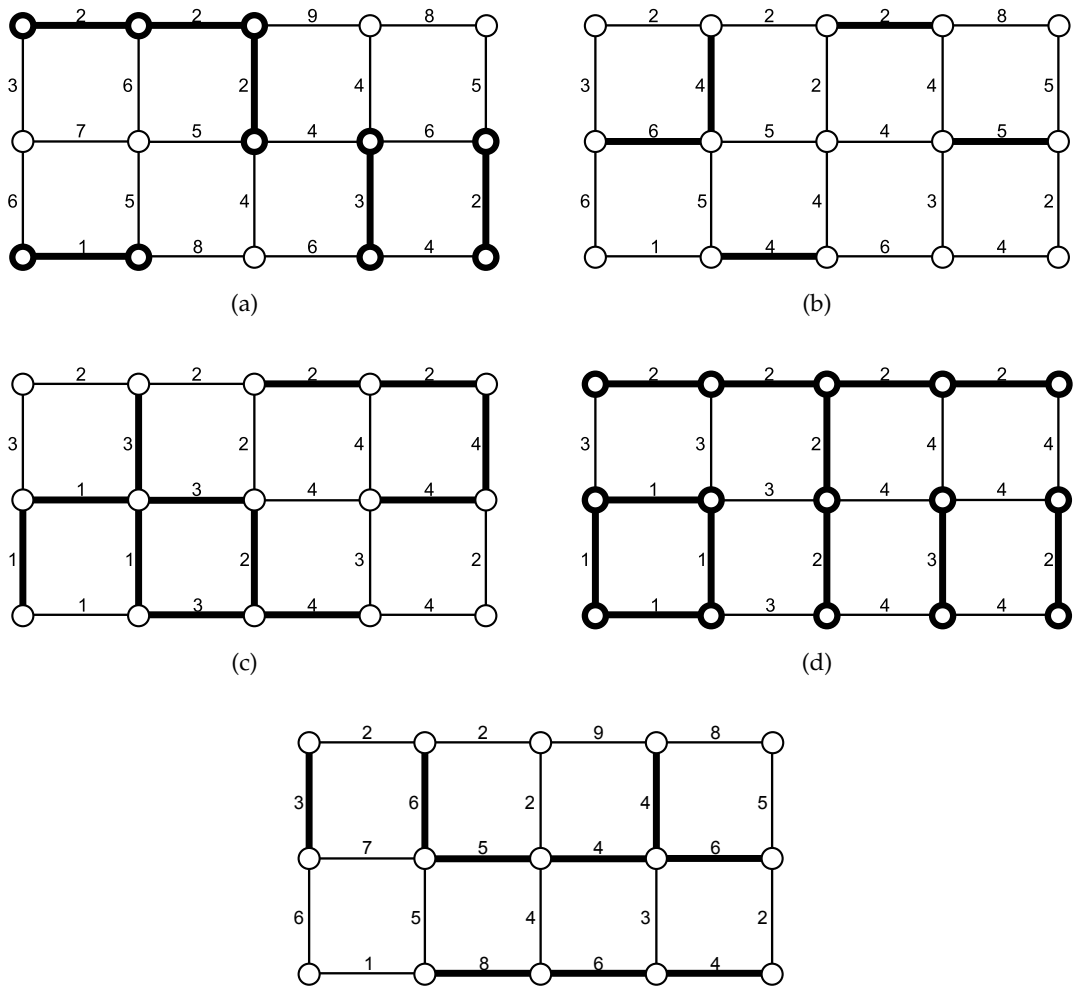


FIGURE 3.13 – Graphe G avec : (a) une application de poids P sur les arêtes et (en gras) $Min(P)$; (b) un ravinement de P et (en gras) les arêtes de poids différents ; (c) un cloisonnement par ravinement de P et (en gras) les arêtes de poids différents ; (d) le cloisonnement par ravinement de P présenté en (c) et (en gras) ses minima régionaux ; (e) l'application de poids P et (en gras) une LPE topologique de G pour P.

Il est à souligner qu'un ravinement est une application de poids (de valeurs inférieures à celles d'origine) sur l'ensemble du graphe, contrairement à l'amincissement par immersion qui donne pour résultat un sous-ensemble des éléments pondérés. Ceci forme la différence majeure entre la LPE par immersion et la LPE topologique puisque dans la première on travaille directement avec un ensemble de sommets que l'on réduit au fur et à mesure alors que dans la seconde, on travaille sur l'ensemble des éléments pondérés de sorte à trouver une nouvelle application de poids dont on déduit, dans notre cas, l'ensemble d'éléments à comparer.

Un exemple de LPE topologique sur une image 2D en niveaux de gris, accompagnée de son cloisonnement ainsi que de représentations topographiques correspondantes aux étapes intermédiaires de ravinement, se trouve dans la figure 3.14.

Remarque 3.21.
Toute LPE topologique est un ensemble séparant mais pas nécessairement un ensemble frontière. De plus, dans un graphe valué, il peut exister plusieurs LPE topologiques distinctes. Enfin, seule la propriété P4 est vérifiée par une LPE topologique.

Des illustrations de la remarque 3.21 se trouvent dans les figures 3.15 et 3.16.

En effet, dans la figure 3.15, pour un graphe et une application de poids identiques, il y a deux LPE topologiques différentes (figures 3.15b et 3.15c). De plus, la LPE topologique de la figure 3.15c n'est pas mince puisque le sommet de poids 7 est intérieur. On note également que, pour la LPE topologique de la figure 3.15b, il n'y a pas, pour P, de chaîne descendante dans un bassin versant allant du sommet de poids 8 de la rangée du haut, qui appartient à la LPE topologique, aux sommets de poids 2 de cette même rangée qui forment le minimum régional d'un bassin versant adjacent.

Dans la figure 3.16, nous nous rendons compte qu'une LPE topologique, en plus de ne pas être nécessairement fine, n'est pas nécessairement non plus un ensemble frontière.

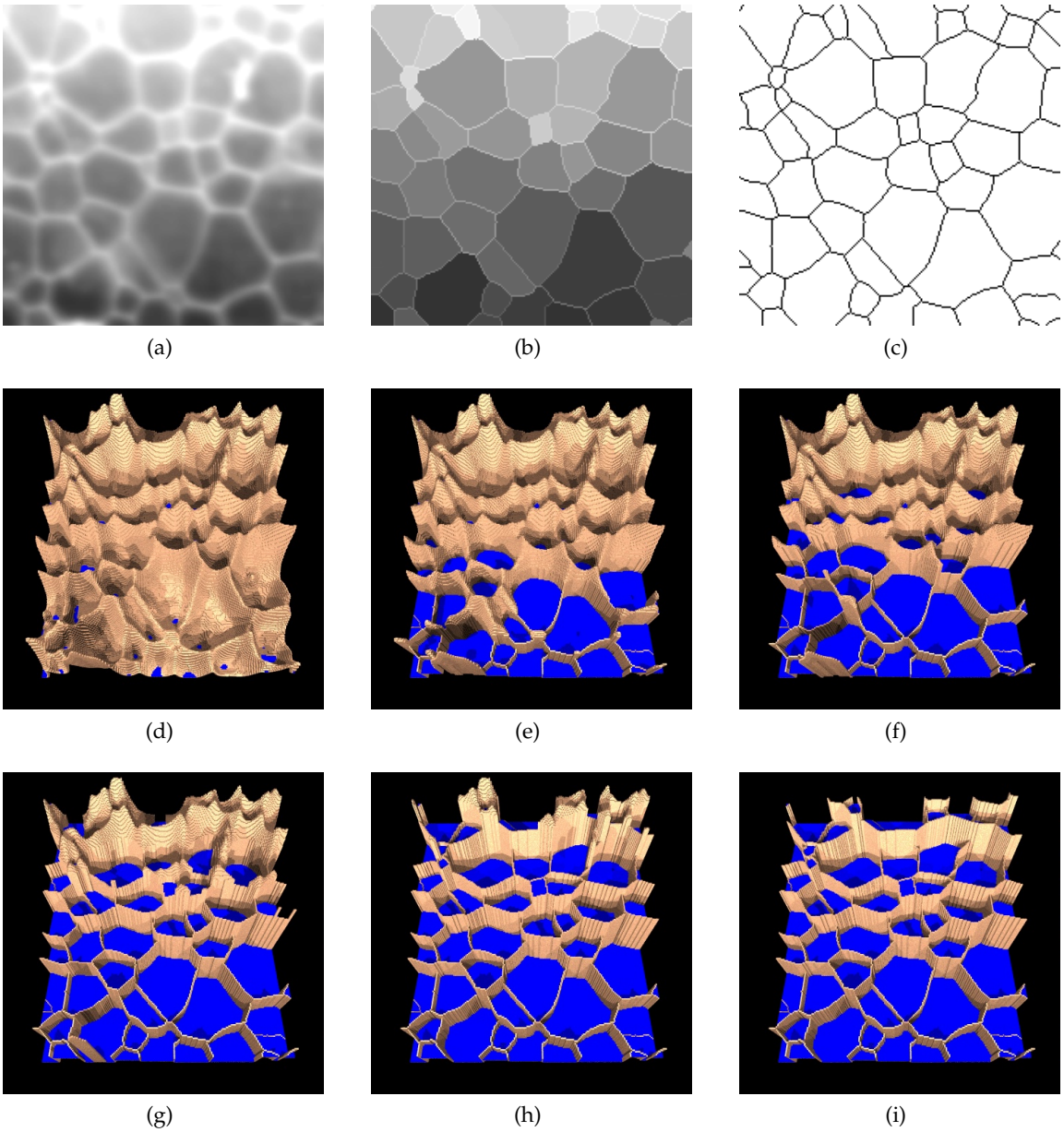
3.2.5 LPE par érosion

La **LPE par érosion** (*C-watershed* en anglais) fut introduite en 2006 dans [59, 60] comme variante de la LPE topologique pour les graphes de fusion parfaite afin de retrouver la propriété d'ensemble frontière induit par le cloisonnement qui n'est pas vérifiée pour cette dernière. Comme pour la LPE topologique, le résultat d'une LPE par érosion est une application de poids. Nous procéderons donc comme dans la section 3.2.4 où nous présenterons le concept de LPE par érosion original mais où nous redéfinirons celle-ci sous forme d'un ensemble frontière déduit de l'application de poids afin de pouvoir comparer cette LPE aux autres. Toutefois, nous mettrons en évidence que la LPE par érosion, dans sa définition originale, est une définition intermédiaire entre la LPE par immersion et la LPE topologique (dans sa définition originale également).

Une fois de plus, il est possible de se représenter le "phénomène physique" engendré par cette méthode :

1. Imaginons une maquette représentant le relief topographique associé à une image.
2. On érode petit à petit les zones situées autour des minima régionaux, en commençant par les plus basses, pour en diminuer l'altitude jusqu'à celle du minimum tout en conservant entre deux minima une séparation dont la hauteur est celle du col le plus bas reliant ces deux minima.

L'ensemble des cloisons ainsi conservées constitue la LPE par érosion de l'image associée.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

FIGURE 3.14 – (a) Une image. (b) Un cloisonnement par ravinement de (a). (c) Une LPE topologique de (a) (liée au cloisonnement de (b)). (c) Représentation topographique de (a) avec en bleu les minima régionaux. (d) à (h) Représentations topographiques d'étapes intermédiaires de ravinement du "relief".

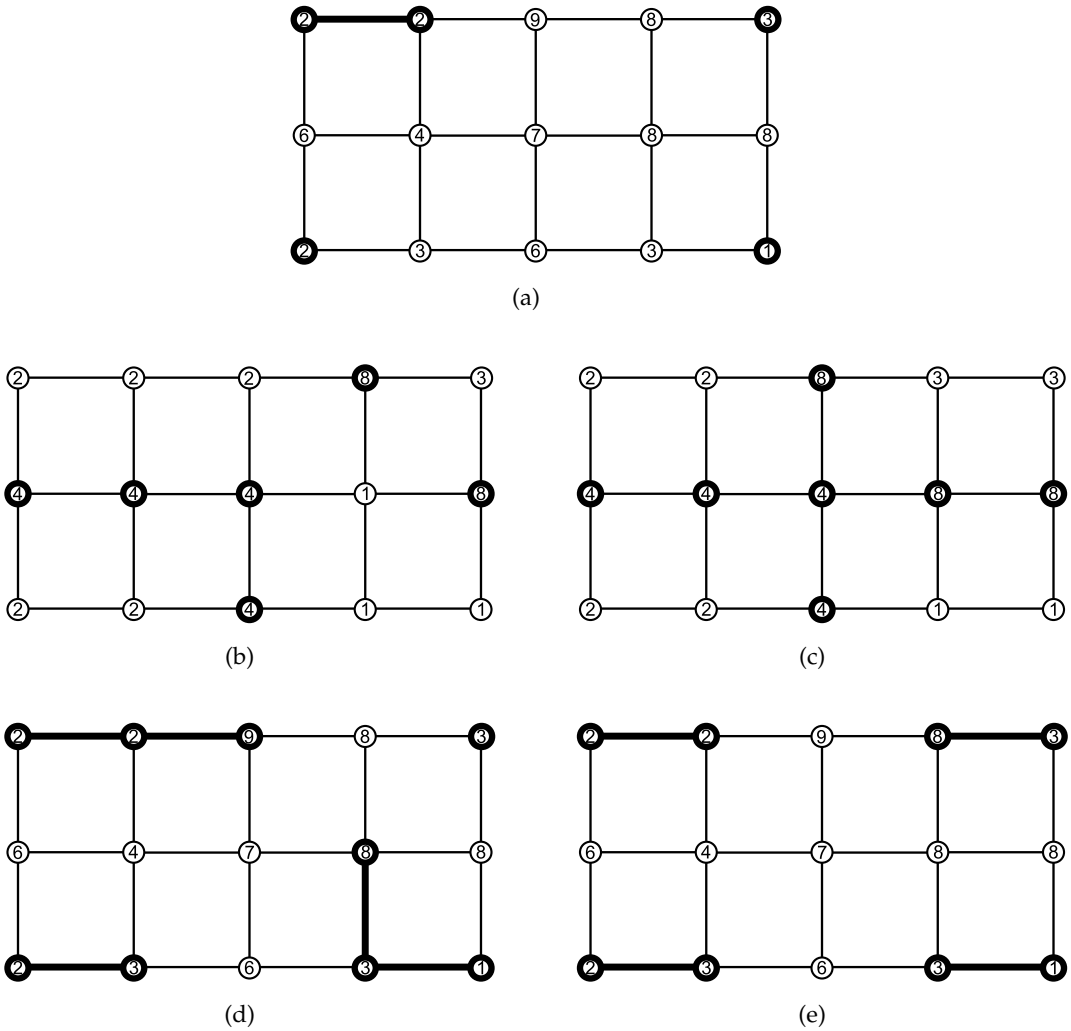


FIGURE 3.15 – Graphe G avec : (a) une application de poids P sur les sommets et (en gras) $Min(G)$; (b,c) un cloisonnement par ravinement de (a) et (en gras) leur LPE topologique ; (d,e) l'application de poids P et (en gras) les bassins versants respectifs des LPEs topologiques des figures (b) et (c).

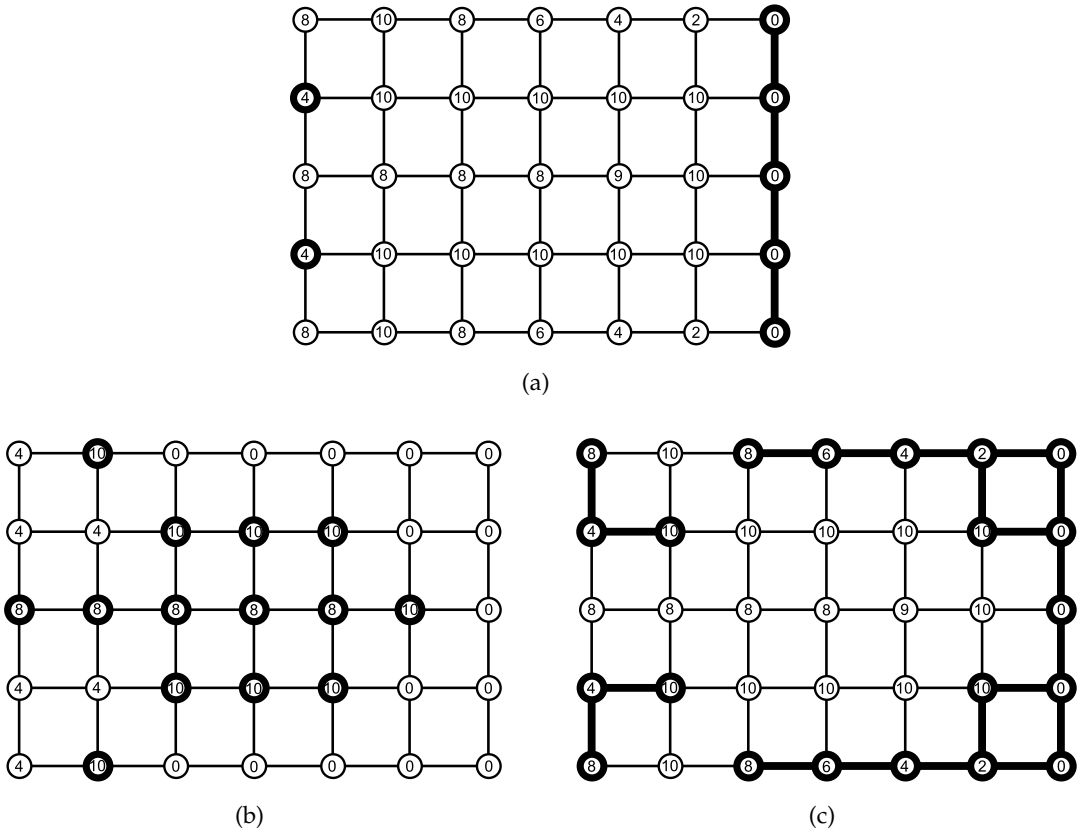


FIGURE 3.16 – Graphe G avec : (a) une application de poids P sur les sommets et (en gras) $Min(G)$; (b) un cloisonnement par ravinement de P et (en gras) sa LPE topologique (qui n'est ni fine, ni un ensemble frontière) ; (c) l'application de poids P et (en gras) les bassins versants respectifs de la LPE topologique de la figure (b).

Une représentation en coupe de ce processus se trouve dans la figure 3.17.

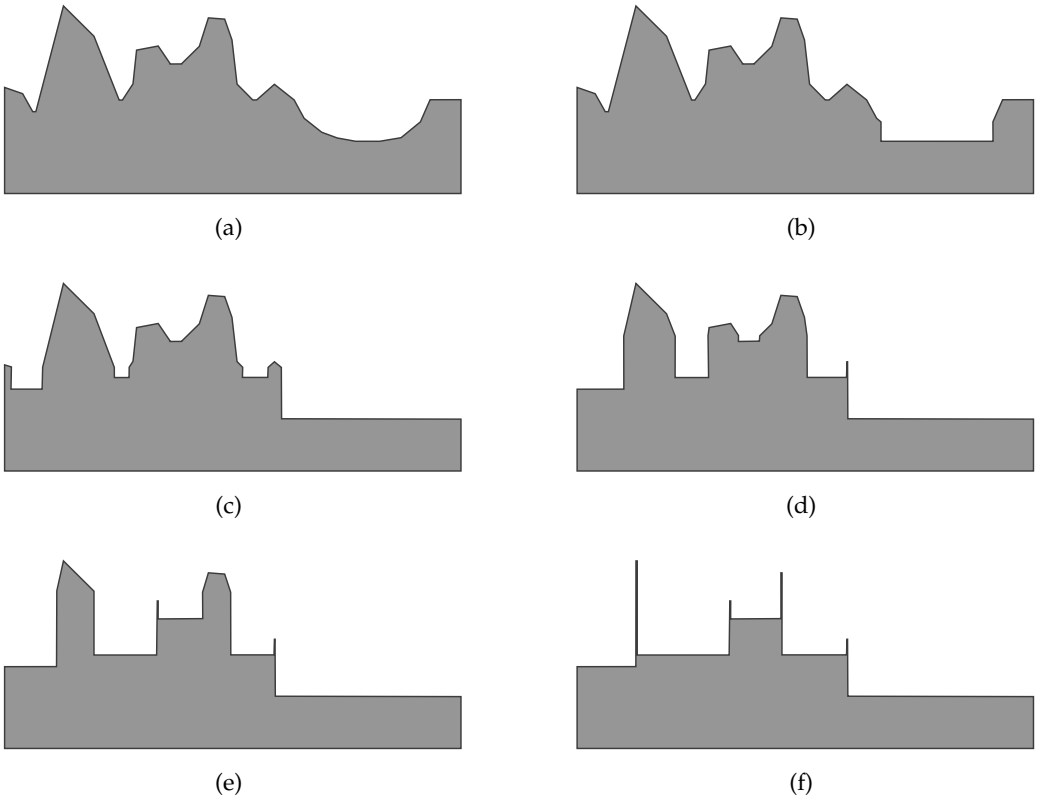


FIGURE 3.17 – (a) Relief topographique associé à une image 1D ; (b) à (e) Erosion de (a) ; (f) Cloisonnement par érosion de (a) dont les "pics" forment une LPE par érosion de (a).

Ce procédé n'est pas sans rappeler la notion d'amincissement présentée en section 2.2 même s'il s'agit ici d'une opération sur l'application de poids et non sur un ensemble d'éléments. En effet, plus l'érosion avance, plus les minima s'étendent et donc plus l'ensemble de séparation diminue. La définition ci-dessous détaille cela.

Définition 3.22 (Erosion)**.**
Soit une application de poids P. Soit E l'ensemble d'éléments de G sur lequel s'effectue la pondération.
On dit qu'une application de poids P' est une ***érosion*** *de P (sur G) si :*
- *$P' = P$, ou*
- *il existe une érosion P'' de P et un élément $e \in E$ tels que :*
  - *e est adjacent à une unique composante connexe M de $Min(P'')$,*
  - *$P'(e) = P''(m)$ avec $m \in E(M)$, et*
  - *$\forall e' \in E$ tel que $e' \neq e$, $P'(e') = P''(e')$.*

Tout comme un ravinement, une érosion ne peut pas créer de nouveaux minima régionaux ni en diminuer l'altitude, cela ne peut que les étendre. Cependant, à la différence d'un ravinement, une érosion ne peut modifier l'altitude que des éléments adjacents à un unique minimum régional.

Remarque 3.23.
Il est à préciser que, dans le cas général, une érosion n'est pas nécessairement un ravinement et réciproquement.

L'érosion correspond à un ravinement guidé pour lequel on abaisse le poids d'un élément destructible (choisi de façon contrainte) au maximum. Il peut néanmoins s'avérer intéressant de contraindre encore davantage le choix de l'élément à abaisser en prenant parmi les éléments adjacents aux minima régionaux celui qui a déjà l'altitude la plus basse comme l'explique la définition d'érosion minimale ci-dessous.

Définition 3.24 (Erosion minimale)**.**
Soit une application de poids P. Soit E l'ensemble d'éléments de G sur lequel s'effectue la pondération.
On dit qu'une application de poids P' est une ***érosion minimale*** *de P (sur G) si :*
- *$P' = P$, ou*
- *il existe une érosion minimale P'' de P et un élément $e \in E$ tels que :*
  - *e est adjacent à une unique composante connexe M de $Min(P'')$,*
  - *$P''(e)$ est minimal,*
  - *$P'(e) = P''(m)$ avec $m \in E(M)$, et*
  - *$\forall e' \in E$ tel que $e' \neq e$, $P'(e') = P''(e')$.*

Malgré la remarque 3.23, tout comme pour le ravinement, un cloisonnement est défini pour l'érosion (minimale ou non).

Définition 3.25 (Cloisonnement par érosion)**.**
Soit une application de poids P et P' une érosion de P.
S'il n'existe aucune érosion à l'application de poids P, alors on dira que P est un ***cloisonnement par érosion****.*
Si P' est un cloisonnement par érosion, on dira que P' est un ***cloisonnement par érosion de*** *P. De plus, si P' est une érosion minimale de P, on dira alors que P' est un* ***cloisonnement par érosion minimale de*** *P.*

Remarque 3.26.
Il est à préciser que la définition de cloisonnement par érosion faite ici correspond en fait à la définition originale de LPE par érosion (C-watershed) faite dans [59, 60].

Des illustrations de ces définitions se trouvent dans les figures 3.18 et 3.19.

La définition de cloisonnement par érosion minimale nous permet ainsi de définir la LPE par érosion ci-dessous.

Définition 3.27 (LPE par érosion)**.**
Soient une application de poids P et P' un cloisonnement par érosion minimale de P. Soit E l'ensemble d'éléments de G sur lequel s'effectue la pondération.
L'ensemble d'éléments de E qui n'appartiennent à aucun minimum régional de G pour P' forme une ***LPE par érosion*** *(de G pour P).*

La définition de LPE par érosion, bien que se rapprochant énormément de celle de LPE topologique, est en fait plus proche de celle de LPE par immersion comme le montre le théorème 3.28 que l'on peut prouver facilement d'après les définitions respectives.

Théorème 3.28.
Soit E l'ensemble d'éléments de G sur lequel s'effectue la pondération. Soit $L \subseteq E$.
L'ensemble d'éléments L est une LPE par immersion si et seulement si L est une LPE par érosion.

Calculer un cloisonnement par érosion peut se faire par un algorithme très similaire à celui d'une LPE de sommets par immersion. Il suffit de baisser itérativement la valeur du sommet adjacent à un unique minimum régional du graphe à l'altitude de ce minimum. On procède ainsi jusqu'à stabilité. La LPE par érosion est l'ensemble de sommets qui, au final, n'appartient à aucun minimum.

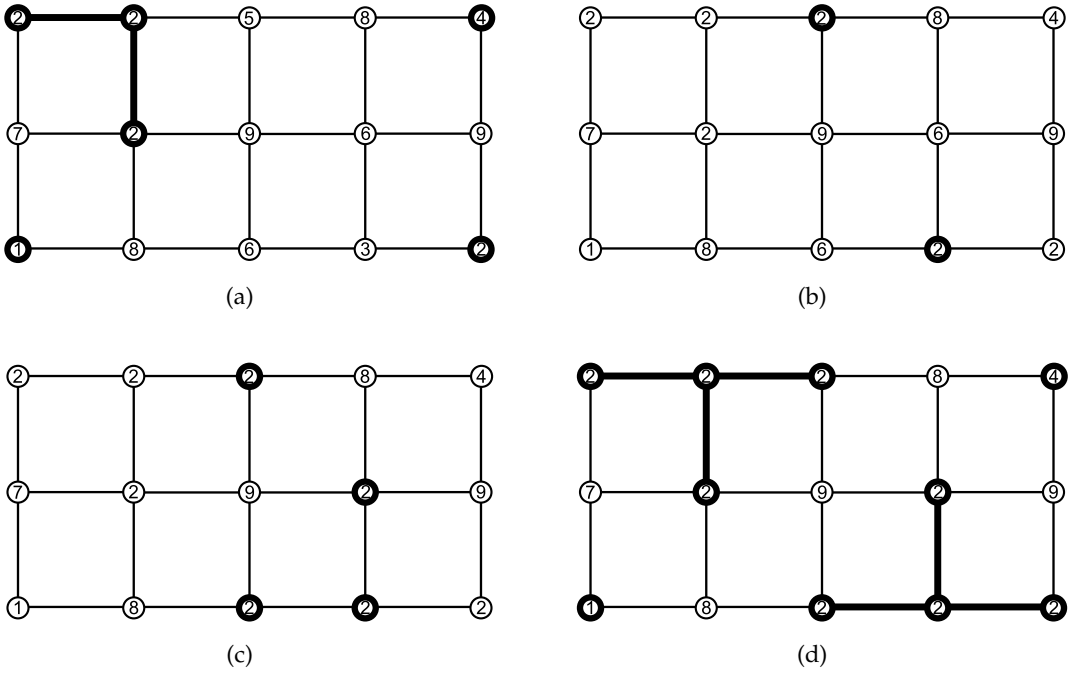


FIGURE 3.18 – Graphe G avec : (a) une application de poids P sur les sommets et (en gras) $Min(P)$; (b) une érosion minimale de P et (en gras) les sommets de poids différents ; (c) un cloisonnement par érosion minimale de P et (en gras) les sommets de poids différents ; (d) le cloisonnement par érosion minimale de P présenté en (c) et (en gras) ses minima régionaux.

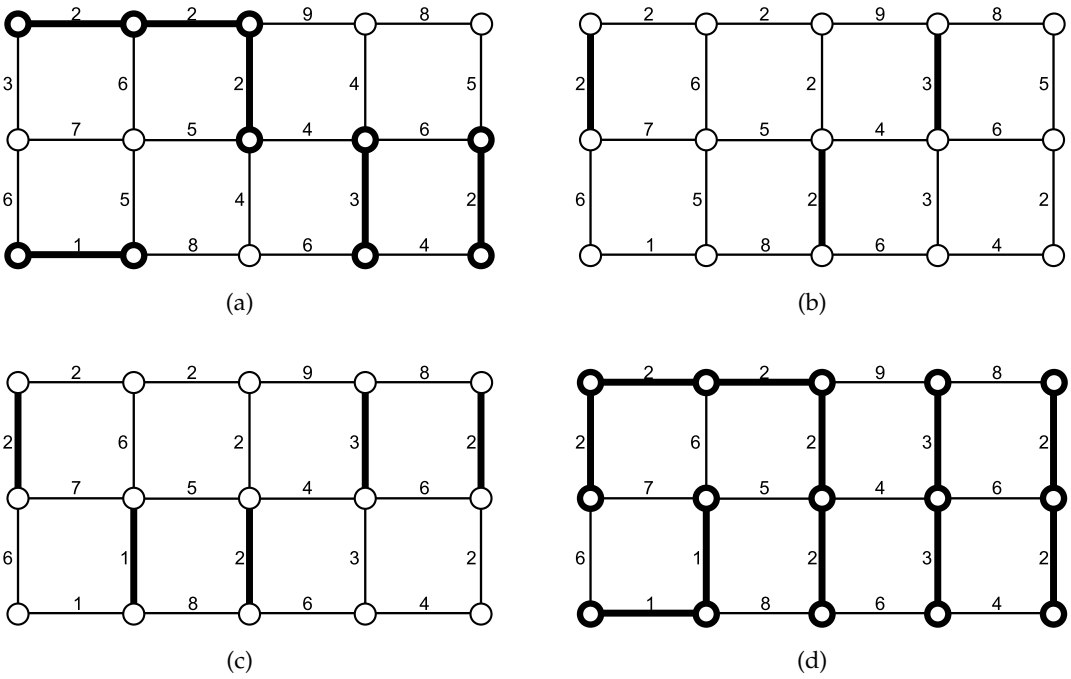


FIGURE 3.19 – Graphe G avec : (a) une application de poids P sur les arêtes et (en gras) $Min(P)$; (b) une érosion minimale de P et (en gras) les arêtes de poids différent ; (c) un cloisonnement par érosion minimale de P et (en gras) les arêtes de poids différent ; (d) le cloisonnement par érosion minimale de P présenté en (c) et (en gras) ses minima régionaux.

Remarque 3.29.
D'après le théorème 3.28, on déduit qu'une LPE par érosion a les mêmes propriétés qu'une LPE par immersion.
Une LPE par érosion est donc un ensemble frontière (de sommets ou d'arêtes) qui n'est pas nécessairement unique. Seule la propriété P1 est donc vérifiée.

Une illustration de la remarque 3.29 se trouve dans la figure 3.20.

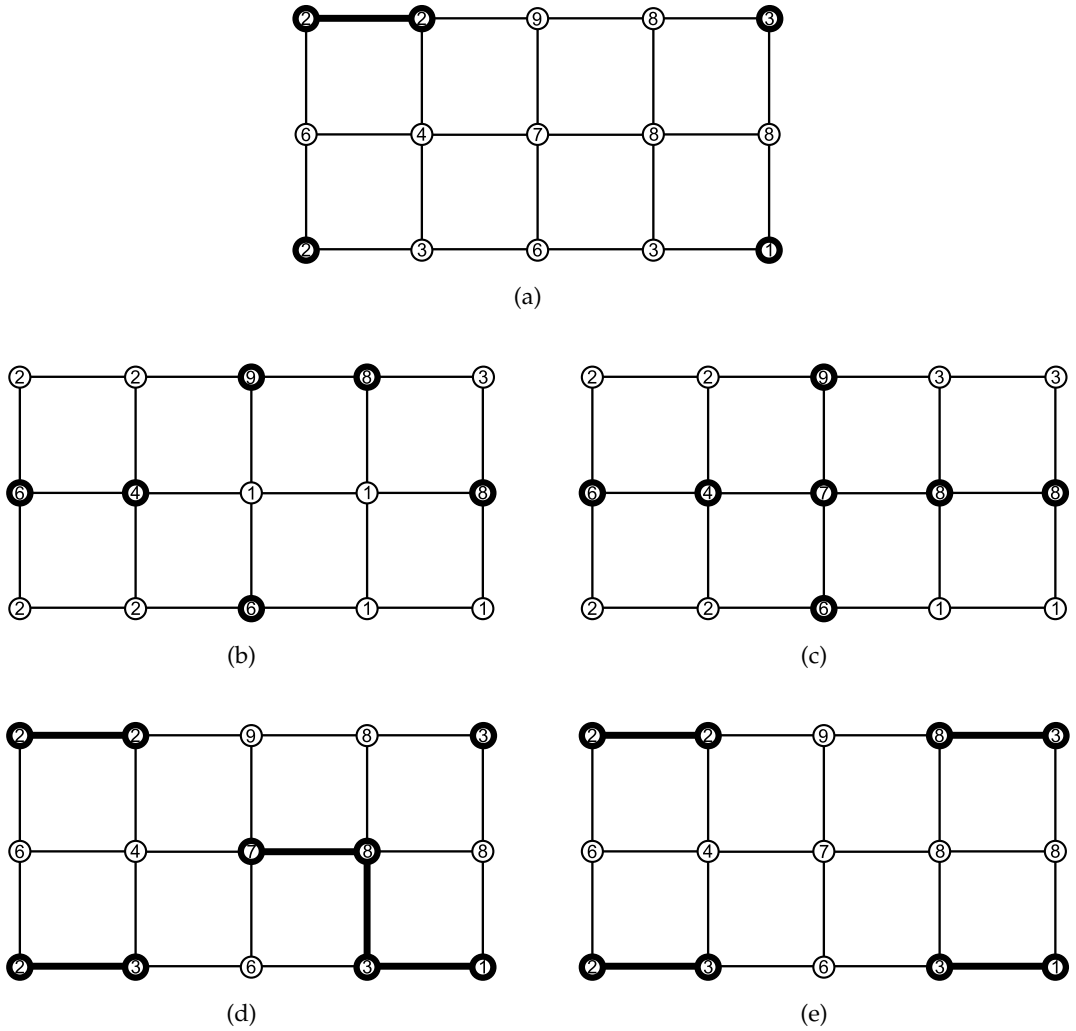


FIGURE 3.20 – Graphe G avec : (a) une application de poids P sur les sommets avec (en gras) $Min(G)$; (b,c) un cloisonnement par érosion de P et (en gras) leur LPE par érosion ; (d,e) l'application de poids P et (en gras) les bassins versants respectifs des LPE topologiques des figures (b) et (c).

3.3 LPE sur graphes à arêtes valuées

Une LPE sur un graphe à arêtes valuées est un ensemble frontière d'arêtes. Comme on l'a déjà vu dans le chapitre 2, de tels ensembles frontières ont l'avantage d'être toujours fins (théorème 2.15).

La définition proposée dans [57, 58] repose sur le phénomène originel du domaine de la topographie qu'est l'écoulement des eaux le long d'une pente, aussi appelé ruissellement, de manière similaire à la LPE par distance topographique. Ce principe consiste

donc à suivre le parcours des gouttes d'eau tombant sur une surface topographique jusqu'à atteindre les minima régionaux.

Le fait de chercher un ensemble frontière d'arêtes, qui est donc fin, permet de formaliser la notion de LPE de manière totalement différente comme le montre la définition 3.30.

Définition 3.30 (LPE d'arêtes)**.**
Soit une application de poids P sur les arêtes.
*On dit que l'ensemble d'arêtes $L \subseteq A$ est une **LPE (d'arêtes)** (de G pour P) si pour toute arête $a \in L$ il existe deux chaînes descendantes d'arêtes $\langle a, x_1, \ldots, x_m \rangle$ et $\langle a, y_1, \ldots, y_n \rangle$ tels que x_m et y_n appartiennent à deux minima régionaux différents.*

Le principe de ruissellement sur lequel repose cette définition fut déjà utilisé pour la LPE par distance topographique qui définissaient non pas un ensemble séparant mais un ensemble de régions. Cette notion peut donc également être interprétée par le biais des bassins versants, comme le montre la définition 3.31.

Définition 3.31 (Partition de bassins versants)**.**
Soit une application de poids P sur les arêtes.
*On dit que l'extension maximale B de $Min(P)$ est une **partition de bassins versants** si, pour tout sommet $s \in S$, il existe dans B une chaîne de plus grande pente de S à $Min(P)$.*

Ces deux définitions semblent donc permettre de définir la LPE soit par sa segmentation, qui est l'ensemble d'arêtes à partir desquels une goutte d'eau peut rejoindre deux minima régionaux différents, soit par sa partition de bassins versants, qui représentent les différentes régions dans lesquelles les gouttes d'eau convergent vers un même minimum régional comme le montre le théorème suivant.

Théorème 3.32 ([57], théorème 1)**.**
Soit une application de poids P sur les arêtes.
L'ensemble d'arêtes $L \subseteq A$ est une LPE d'arêtes (de G pour P) si et seulement si L est la coupe induite par une partition de bassins versants.

Le théorème 3.32 montre que les graphes à arêtes valuées forment donc un cadre particulièrement bien adapté à la LPE puisqu'il permet l'équivalence entre ces deux formulations, ce qui n'est le cas pour aucune autre des définitions de LPE vues jusqu'ici.

Remarque 3.33.
Il est à noter qu'une partition de bassins versants est une extension (au sens de la définition 2.1) de la LPE par distance topographique (dans le cas des graphes à arêtes valuées) puisque, comme on l'a vu, cette dernière n'est pas nécessairement une partition des éléments du graphe, mettant ainsi en évidence le lien entre LPE d'arêtes LPE par distance topographique.

Remarque 3.34.
Toute LPE d'arêtes est un ensemble frontière d'arêtes.
De plus, dans un graphe à arêtes valuées, il peut exister plusieurs LPE d'arêtes distinctes. Une LPE d'arêtes satisfait les propriétés P1 à P5.

Une illustration de la remarque 3.34 se trouve dans la figure 3.21.

La suite de cette section sera consacrée aux définitions qui nous seront utiles pour la comparaison des différents paradigmes de LPE mais également à décrire un des deux algorithmes linéaires de calcul de LPE d'arêtes présentés dans [58]. Nous commençons par définir une catégorisation des arêtes d'un graphe afin de les classifier selon leur "position topographique".

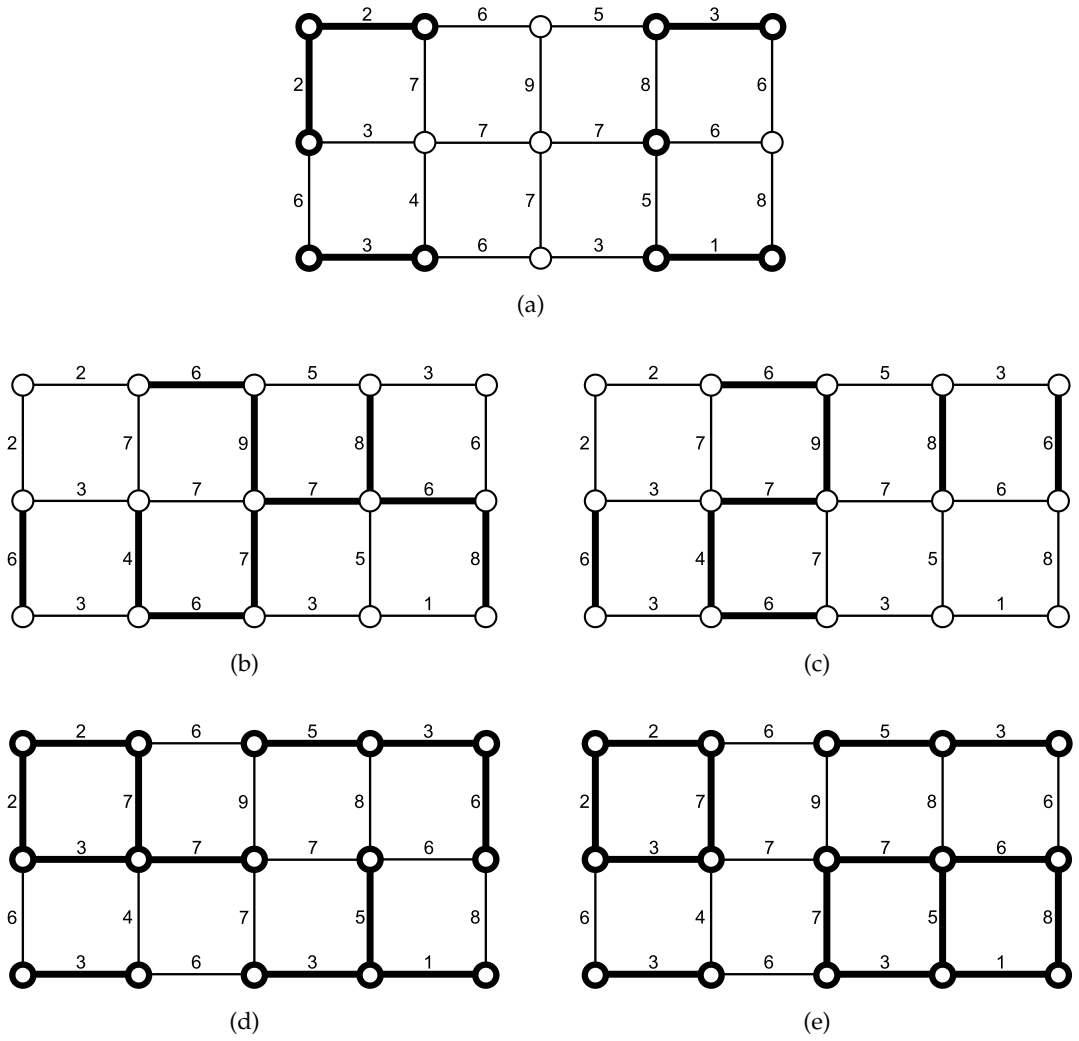


FIGURE 3.21 – Graphe G et application de poids P sur les arêtes avec (en gras) : (a) $Min(G)$; (b,c) deux LPE ; (d,e) les bassins versants respectifs des LPEs des figures (b) et (c).

Définition 3.35 (Catégories d'arêtes)**.**
Soit une application de poids P sur les arêtes.
On dit que l'arête $u = \{x, y\} \in A$ est :
- *une **arête intérieure** (pour P) si toutes les arêtes adjacentes à u ont une altitude supérieure ou égale à celle de u ;*
- *une **arête séparante** (pour P) si, pour chaque sommet extrémité de u, au moins une arête incidente à ce sommet a une altitude strictement inférieure à celle de u ;*
- *une **arête de bord** (pour P) sinon (autrement dit, un sommet extrémité de u a au moins une arête incidente d'altitude strictement inférieure à celle de u et son autre extrémité a toutes ses arêtes incidentes d'altitude supérieure ou égale à celle de u).*

Remarque 3.36.
Soient une application de poids P_A sur les arêtes et P_S sur les sommets tels que à tout sommet $s \in S$ on attribue le poids $P_S(s) = \min\{P_A(a) \mid s \in a\}$.
Ainsi, une arête $a = \{x, y\}$ est :
- *intérieure (pour P_A) si $P_A(a) = P_S(x) = P_S(y)$;*
- *séparante (pour P_A) si $P_A(a) > \max\{P_S(x), P_S(y)\}$;*
- *de bord (pour P_A) si $P_A(a) = \max\{P_S(x), P_S(y)\}$ et $P_A(a) > \min\{P_S(x), P_S(y)\}$.*

La définition d'arêtes de bord va nous permettre de définir un nouveau type de ravinement ainsi que son cloisonnement comme vont le montrer les définitions suivantes.

Définition 3.37 (Ravinement de bord)**.**
Soit une application de poids P sur les arêtes.
*On dit qu'une application de poids P' est un **ravinement de bord** de P (sur G) si :*
- *$P' = P$, ou*
- *il existe un ravinement par bord P'' de P et une arête $u \in A$ tels que :*
  - *u est une arête de bord,*
  - *$P'(u) = \min\{P(u') \mid u'$ est adjacente à $u\}$, et*
  - *$\forall u' \in A$ tel que $u' \neq u$, $P'(u') = P''(u')$.*

Il découle tout naturellement de la définition de ravinement de bord celle du cloisonnement correspondant.

Définition 3.38 (Cloisonnement par ravinement de bord)**.**
Soit une application de poids P et P' un ravinement de bord de P.
*S'il n'existe aucun ravinement de bord à l'application de poids P, alors on dira que P est un **cloisonnement par ravinement de bord**.*
*De plus, si P' est un cloisonnement par ravinement de bord, on dira que P' est un **cloisonnement par ravinement de bord de** P.*

Nous passons maintenant aux érosions de bord, qui sont des ravinements de bord auxquels on rajoute une condition sur l'arête à abaisser qui doit alors être adjacente aux minima régionaux. Cependant, la propriété 3.39 ci-dessous nous permet de nous rendre compte qu'une telle arête est en fait strictement adjacente aux minima régionaux.

Propriété 3.39.
Soit une application de poids P sur les arêtes.
Si $a \in A$ est une arête de bord adjacente à $Min(P)$, alors a est une arête strictement adjacente à $Min(P)$.

Preuve de la propriété 3.39.
Considérons que que l'arête $a \in A$ est une arête de bord adjacente à $Min(P)$ mais pas strictement adjacente. Cela implique donc que les deux sommets extrémités de a appartiennent à $Min(P)$. En conséquence de quoi, soit a est une arête séparante, soit $a \in A(Min(P))$, ce qui est impossible. L'arête a est donc bien strictement adjacente à $Min(P)$. □

On obtient donc la définition ci-dessous.

Définition 3.40 (Erosion de bord)**.**
Soit une application de poids P sur les arêtes.
On dit qu'une application de poids P' est une ***érosion de bord*** *de P (sur G) si :*
- *$P' = P$, ou*
- *il existe une érosion de bord P'' de P et une arête $u \in A$ tels que :*
  - *u est une arête de bord,*
  - *u est (strictement) adjacente à $Min(P'')$,*
  - *$P'(u) = \min\{P(u') \mid u' \text{ est adjacente à } u\}$, et*
  - *$\forall u' \in A$ tel que $u' \neq u$, $P'(u') = P''(u')$.*

Là encore on définit le cloisonnement correspond comme ci-dessous.

Définition 3.41 (Cloisonnement par érosion de bord)**.**
Soit une application de poids P et P' une érosion de bord de P.
S'il n'existe aucune érosion de bord à l'application de poids P, alors on dira que P est un ***cloisonnement par érosion de bord****.*
De plus, si P' est un cloisonnement par érosion de bord, on dira que P' est un ***cloisonnement par érosion de bord de*** *P.*

De manière similaire à la section 3.2.5, nous allons maintenant définir l'érosion minimale de bord à partir de l'érosion de bord. Celle-ci est donc une érosion de bord à laquelle on ajoute un ordre de priorité de l'abaissement des altitudes des arêtes selon l'ordre croissant de leur poids initial. Toutefois, la propriété 3.42 ci-dessous nous permet de nous rendre compte que la notion d'arête de bord n'est pas indispensable dans la définition d'érosion minimale de bord.

Propriété 3.42.
Soit une application de poids P sur les arêtes.
L'arête $a \in A$ est une arête de bord de poids minimal si et seulement si a est une arête strictement adjacente à $Min(P)$ de poids minimal.

Preuve de la propriété 3.42.
- Soit $a \in A$ une arête de bord de poids minimal. L'arête a est donc adjacente à $Min(P)$. On déduit de la propriété 3.39 que a est strictement adjacent à $Min(P)$. Considérons maintenant que l'arête a' strictement adjacente à $Min(P)$ de poids minimal ne soit pas une arête de bord. On a donc $P(a') < P(a)$. L'arête a' étant adjacente à $Min(P)$, elle n'est donc pas une arête intérieure. L'arête a' est donc une arête séparante. Or puisque a' est de poids le plus faible, cela signifie que les deux sommets extrémités de a' sont inclus dans $Min(P)$. Auquel cas a' n'est pas strictement adjacente à $Min(P)$. Ce qui contredit l'hypothèse. Si l'arête a est une arête de bord de poids minimale alors a est une arête minimale strictement adjacente à $Min(P)$.
- Soit $a \in A$ une arête strictement adjacente à $Min(P)$ de poids minimal. Considérons que a ne soit pas une arête de bord. L'arête a étant adjacente à $Min(P)$, elle n'est donc pas une arête intérieure. L'arête a est donc une arête séparante. Or puisque a est de poids le plus faible, cela signifie que les deux sommets extrémités de a sont inclus dans $Min(P)$. Auquel cas a n'est pas strictement adjacente à $Min(P)$. Ce qui contredit l'hypothèse. L'arête a est donc bien une arête de bord. En conséquence de quoi, si a est une arête strictement adjacente à $Min(P)$ de poids minimal, alors a est une arête de bord de poids minimal. □

Outre le fait que la propriété 3.42 nous permette d'obtenir la définition suivante sans mentionner d'arêtes de bord, il nous indique que la condition d'adjacence aux minima régionaux est une conséquence induite par la sélection des arêtes de bord à abaisser se fait par ordre croissant.

Définition 3.43 (Erosion minimale de bord)**.**
Soit une application de poids P sur les arêtes.
On dit qu'une application de poids P' est une ***érosion minimale de bord*** *de P (sur G) si :*
- *$P' = P$, ou*
- *il existe une érosion minimale de bord P'' de P et une arête $u \in A$ tels que :*
  - *u est strictement adjacente à $Min(P'')$,*
  - *$P''(u)$ est minimal,*
  - *$P'(u) = \min\{P(u') \mid u' \text{ est adjacente à } u\}$, et*
  - *$\forall u' \in A$ tel que $u' \neq u$, $P'(u') = P''(u')$.*

Il est à noter que, grâce à la propriété 3.42, la notion d'arête de bord n'a pas besoin d'apparaître dans la définition 3.43. On déduit de cette définition le cloisonnement correspondant ci-après.

Définition 3.44 (Cloisonnement par érosion minimale de bord)**.**
Soit une application de poids P et P' une érosion minimale de bord de P.
S'il n'existe aucune érosion (minimale) de bord à l'application de poids P, alors on dira que P est un ***cloisonnement par érosion minimale de bord****.*
De plus, si P' est un cloisonnement par érosion minimale de bord, on dira que P' est un ***cloisonnement par érosion minimale de bord de*** *P.*

Même si les notions de ravinement de bord, érosion de bord et érosion minimale de bord s'incluent l'une dans l'autre comme expliqué au fur et à mesure de leurs définitions, elles n'en restent pas moins différentes. Il n'en est cependant pas de même pour leurs cloisonnements respectifs comme le montre le théorème 3.45.

Théorème 3.45.
Soient P et P' deux applications de poids sur les arêtes.
Les trois propositions suivantes sont équivalentes :
- *P' est un cloisonnement par ravinement de bord de P ;*
- *P' est un cloisonnement par érosion de bord de P ;*
- *P' est un cloisonnement par érosion minimale de bord de P.*

Pour regrouper ces trois notions équivalentes nous parlerons dorénavant de ***cloisonnement par bord****.*

C'est à travers cette équivalence que nous mettons en évidence le lien qui unit ces définitions au cadre de LPE d'arêtes défini dans cette section comme le montre le théorème 3.46.

Théorème 3.46.
Soit une application de poids P sur les arêtes.
L'ensemble d'arêtes $L \subseteq A$ est une LPE d'arêtes si et seulement si L est la coupe induite par les minima régionaux d'un cloisonnement par bord.

Le théorème 3.46 nous permet donc d'utiliser la définition de n'importe quel cloisonnement par bord pour déterminer une LPE d'arêtes dans un graphe à arêtes valuées. C'est celle de cloisonnement par érosion de bord qui permet d'obtenir l'algorithme linéaire suivant avec une application de poids P sur les arêtes. Soit l'application de poids P' sur les arêtes qui est une copie de P

1. Calculer $Min(P')$.
2. Prendre une arête strictement adjacente à un minimum régional $M \in Min(P')$ qui soit une arête de bord et diminuer son altitude jusqu'à celle de M.
3. Répéter l'étape précédente jusqu'à idempotence.

L'application de poids P' résultant est un cloisonnement par bord de P.

Dans [57, 58] est décrit un autre algorithme linéaire permettant d'obtenir une LPE sur graphes à arêtes valuées. Celui-ci, qui n'a pas besoin d'extraire les minima régionaux du graphe (contrairement à celui détaillé ci-dessus), se base sur les chaînes de plus grande pente et leur fusion qui sont interprétés en termes de courants.

3.4 LPE relative

Quel que soit le type de LPE considéré, la plupart du temps, lorsque l'on désire calculer une LPE, le résultat obtenu est sur-segmenté. En effet, puisque les bassins versants d'une LPE forment une partition qui est une extension maximale des minima régionaux, on obtient donc autant de bassins versants que de minima régionaux (qui peuvent être très nombreux dans une image bruitée par exemple).

Deux possibilités existent pour remédier à cela :

- un pré-traitement au calcul de la LPE visant à diminuer le nombre de minima régionaux de l'application de poids (lissage,...) ;
- un post-traitement au calcul de la LPE visant à regrouper certains bassins versants entre eux (fusion de régions,...).

Ces méthodes permettent ainsi de limiter le nombre de régions de la segmentation selon des critères propres à l'application dans laquelle elle intervient. Il est néanmoins parfois souhaitable de pouvoir déterminer manuellement les marqueurs initiaux à la segmentation. Ceci permet de fixer dès le début le nombre de régions de la segmentation tout en "orientant" son résultat en initialisant ainsi certains sommets dans une région précise.

Ce type de segmentation reposant sur des marqueurs et utilisant un paradigme de LPE s'appelle **LPE relative**.

Pour ce faire, un post-traitement, proposé dans [145], consiste à construire un graphe représentant le voisinage des bassins versants en pondérant les arêtes avec l'altitude de connexion entre les bassins versants voisins. Les sommets représentant les bassins versants dans lesquels se trouvent un marqueur sont étiquetés comme tels. Enfin, une forêt de poids minimum (voir chapitre 4) est calculée sur ce graphe. Le partitionnement des sommets est reporté sur chaque bassin versant correspondant pour ainsi obtenir la segmentation désirée.

Cependant, cette méthode ne traite pas le cas où des marqueurs différents appartiennent à un même bassin versant de la première LPE.

Une alternative, présentée dans [148, 206], souvent préférée compte tenu de la restriction soulignée précédemment, est de procéder à une reconstruction morphologique de l'application de poids avant de procéder au calcul de la LPE. Cette reconstruction permet de créer, à partir de l'application de poids P et d'un sous-graphe M, une application de poids P' (sur le même type d'élément que P) telle que M soit l'ensemble des minima régionaux de P'. On calcule alors la LPE sur P' de manière habituelle.

La suite de cette section sera consacrée à la description de cette reconstruction. Nous y décrirons le principe d'inondation (introduit dans [147, 186]), d'inondation contrainte et, enfin celui d'inondation restreinte qui nous permettra finalement de définir précisément une LPE relative.

Par la suite, nous considérerons que le sous-graphe M contient au moins deux composantes connexes afin d'avoir au moins deux minima régionaux dans l'application de poids reconstruite.

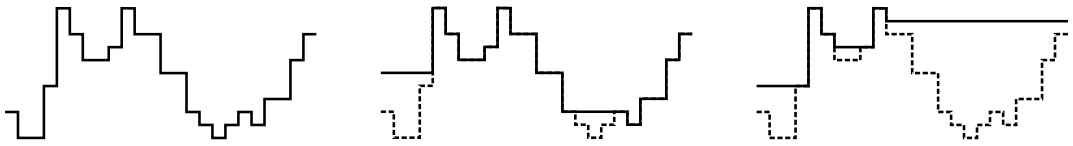
Une fois de plus, on peut se représenter le "phénomène physique" engendré par cette méthode :

1. Imaginons une maquette représentant le relief topographique associé à une image.
2. Comme son nom l'indique, une inondation, consiste à verser de l'eau dans un ou plusieurs bassins de ce relief pour ainsi obtenir une surface plus élevée. Ce "remplissage" n'a pas à être du même niveau pour tous les bassins.

La définition 3.47 précise cette notion dans le cadre des graphes.

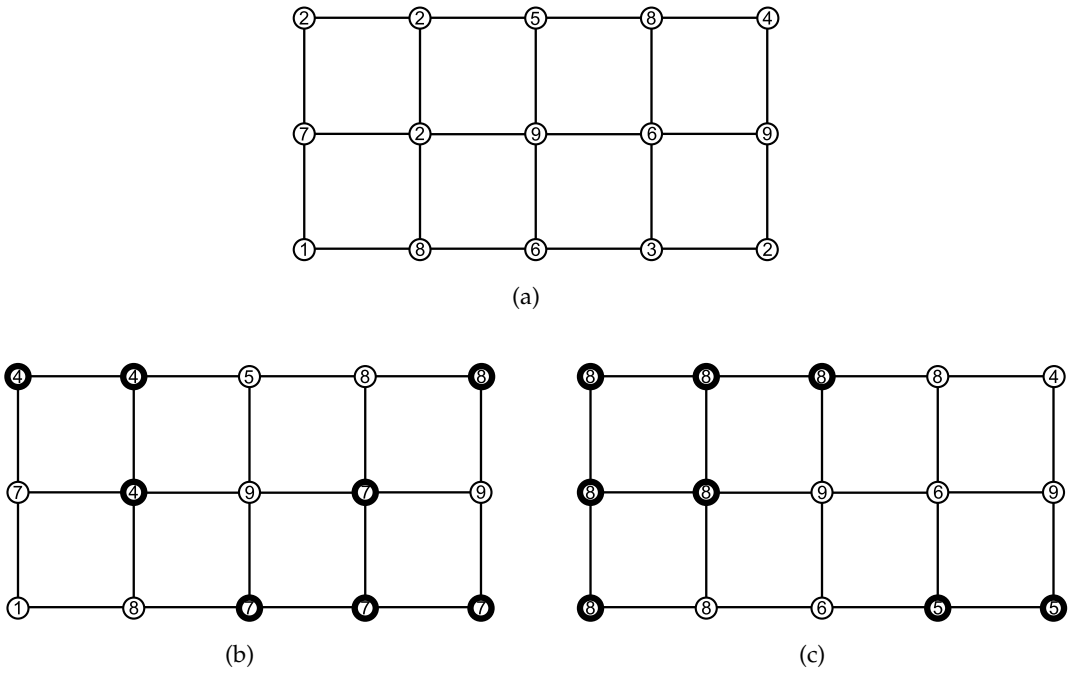
Définition 3.47 (Inondation)**.**
Soit une application de poids P.
On dit que l'application de poids P' est une ***inondation*** *de P (sur G) si, pour tout $h \in \mathbb{R}$, toute composante connexe du seuillage inférieur $G'^{h}_{P'}$ est une composante connexe du seuillage inférieur G^{h}_{P}.*
On note $\mathcal{F}(P)$ ($\mathcal{F}$ pour flooding*, inondation en anglais) l'ensemble des inondations de P.*

Des illustrations de cette définition se trouvent dans les figures 3.22, 3.23 et 3.24.



(a) (b) (c)

FIGURE 3.22 – Représentation 1D de : (a) une application de poids P ; (b,c) une inondation de P (avec P en pointillés).



(a)

(b) (c)

FIGURE 3.23 – Graphe G avec : (a) une application de poids P sur les sommets ; (b,c) une inondation de P et (en gras) les sommets de poids différents.

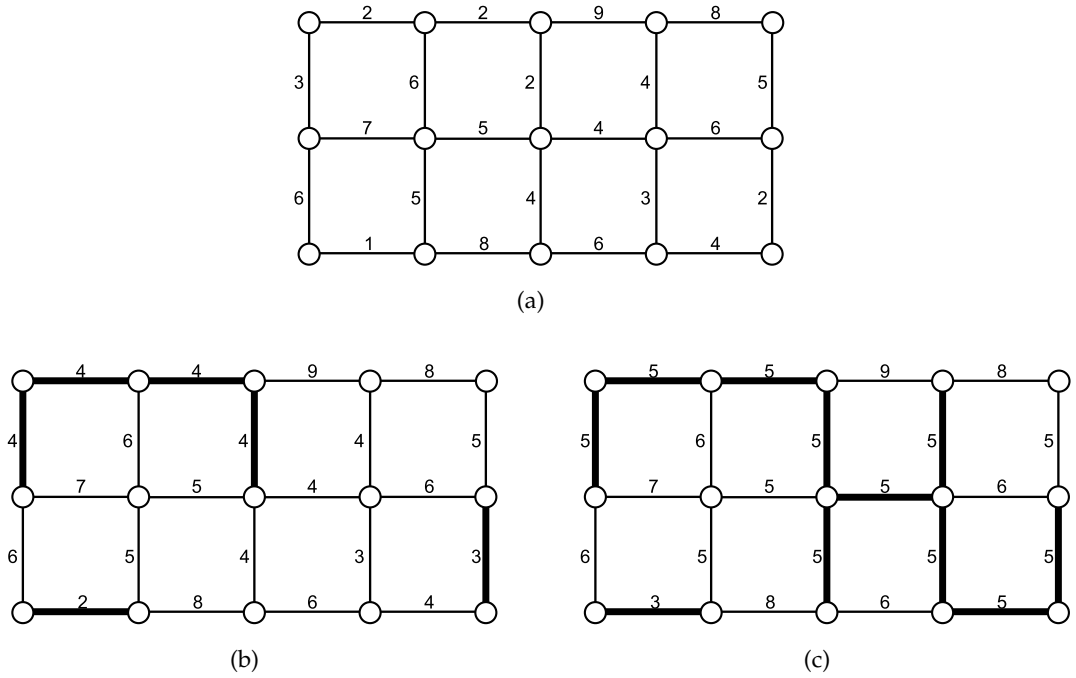


FIGURE 3.24 – Graphe G avec : (a) une application de poids P sur les arêtes ; (b,c) une inondation de P et (en gras) les arêtes de poids différents.

Remarque 3.48.
Il est à noter que, en considérant E comme l'ensemble d'éléments de G sur lequel s'effectue la pondération, par définition d'une inondation P' de P (sur G), nous avons $\forall e \in E, P'(e) \geq P(e)$.

L'inondation nous permet de donner les définitions suivantes.

Définition 3.49 (Inondation contrainte)**.**
Soit une application de poids P. Soit E l'ensemble d'éléments de G sur lequel s'effectue la pondération. Soit un sous-graphe de G.
*On dit que l'application de poids P' est une **inondation de** P **contrainte par** M (sur G) si :*
- *P' est une inondation de P ($P' \in \mathcal{F}(P)$), et*
- *$\forall e \in E(M), P(e) = P'(e)$.*

On note $\mathcal{F}(P, M)$ l'ensemble des inondations de P contraintes par M.

Remarque 3.50.
Il est à noter que l'inondation d'une application de poids P contrainte par ses minima régionaux ($Min(P)$) donne pour résultat P.

Définition 3.51 (Inondation restreinte)**.**
Soit une application de poids P. Soit E l'ensemble d'éléments de G sur lequel s'effectue la pondération. Soit un sous-graphe de G.
Soit l'application de poids P' telle que :

$$\forall e \in E, P'(e) = \begin{cases} \alpha & si\ e \in E(M) \\ P(e) & sinon \end{cases} \tag{3.1}$$

avec $\alpha \leq \min_{e \in E}\{P(e)\}$.
*On dit que l'application de poids P'' est une **inondation de** P **restreinte par** M (sur G) si P'' est une inondation de P' contrainte par M.*
On note $\tilde{\mathcal{F}}(P, M)$ l'ensemble des inondations de P restreintes par M.

Remarque 3.52.
Il est à noter que, si l'on excepte le changement de valeur des poids des éléments de M, l'inondation restreinte contraint les valeurs des éléments adjacents à M à demeurer inchangés par l'inondation alors que ce n'est pas le cas pour une inondation contrainte.

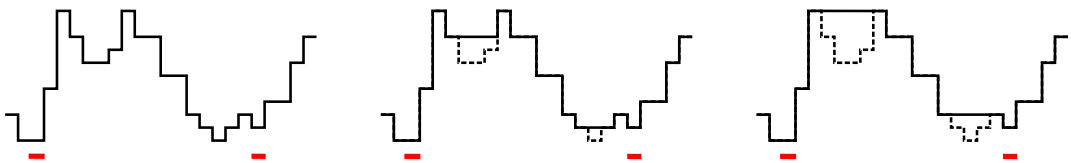
Nous présentons maintenant la définition d'une inondation contrainte ou restreinte spécifique.

Définition 3.53 (Inondation maximale)**.**
Soit une application de poids P. Soit M un sous-graphe de G.
On dit que l'application de poids P′ est l'unique ***inondation maximale de*** *P* ***contrainte, respectivement restreinte, par*** *M (sur G) si :*
- *P′ est une inondation de P contrainte, respectivement restreinte, par M, et*
- *il n'existe pas d'application de poids P″ différente de P′ qui soit une inondation de P′ contrainte par M.*

Remarque 3.54.
Il est à noter que dans le cas où l'inondation n'est pas contrainte/restreinte, parler d'inondation maximale n'a pas de sens puisqu'il n'y a alors pas de limite à la montée du niveau de l'eau.

Remarque 3.55.
Les seuls minima régionaux qui résultent de l'inondation restreinte maximale sont les composantes connexes de M, ce qui rejoint la remarque 3.50.

Des illustrations de ces définitions se trouvent dans les figures 3.25, 3.26, 3.27 et 3.28.



(a) (b) (c)

FIGURE 3.25 – Représentation 1D de : (a) une application de poids P avec un ensemble de marqueurs M (en rouge) ; (b) une inondation de P contrainte par M (avec P en pointillés) ; (c) une inondation maximale de P contrainte par M (avec P en pointillés).

Maintenant que nous avons présenté les notions concernant les inondations, notamment l'inondation restreinte maximale, nous pouvons définir une LPE relative. En effet, comme mentionné dans la remarque 3.55, l'inondation restreinte maximale vise à transformer l'application de poids de telle sorte que les composantes connexes de M deviennent les seuls minima régionaux, permettant ainsi d'employer directement un algorithme traditionnel de LPE.

Définition 3.56 (LPE relative)**.**
Soit une application de poids P. Soit E l'ensemble d'éléments de G sur lequel s'effectue la pondération. Soit M un sous-graphe de G. Soit P′ l'inondation maximale de P restreinte par M ($P' = \tilde{\mathcal{F}}(P, M)$).
On dit que l'ensemble d'éléments $L \subseteq E$ *est une* LPE relative à M (pour P) *si L est une LPE pour* $\tilde{P}$.

Des illustrations de la définition 3.56 se trouvent dans les figures 3.28 et 3.29.

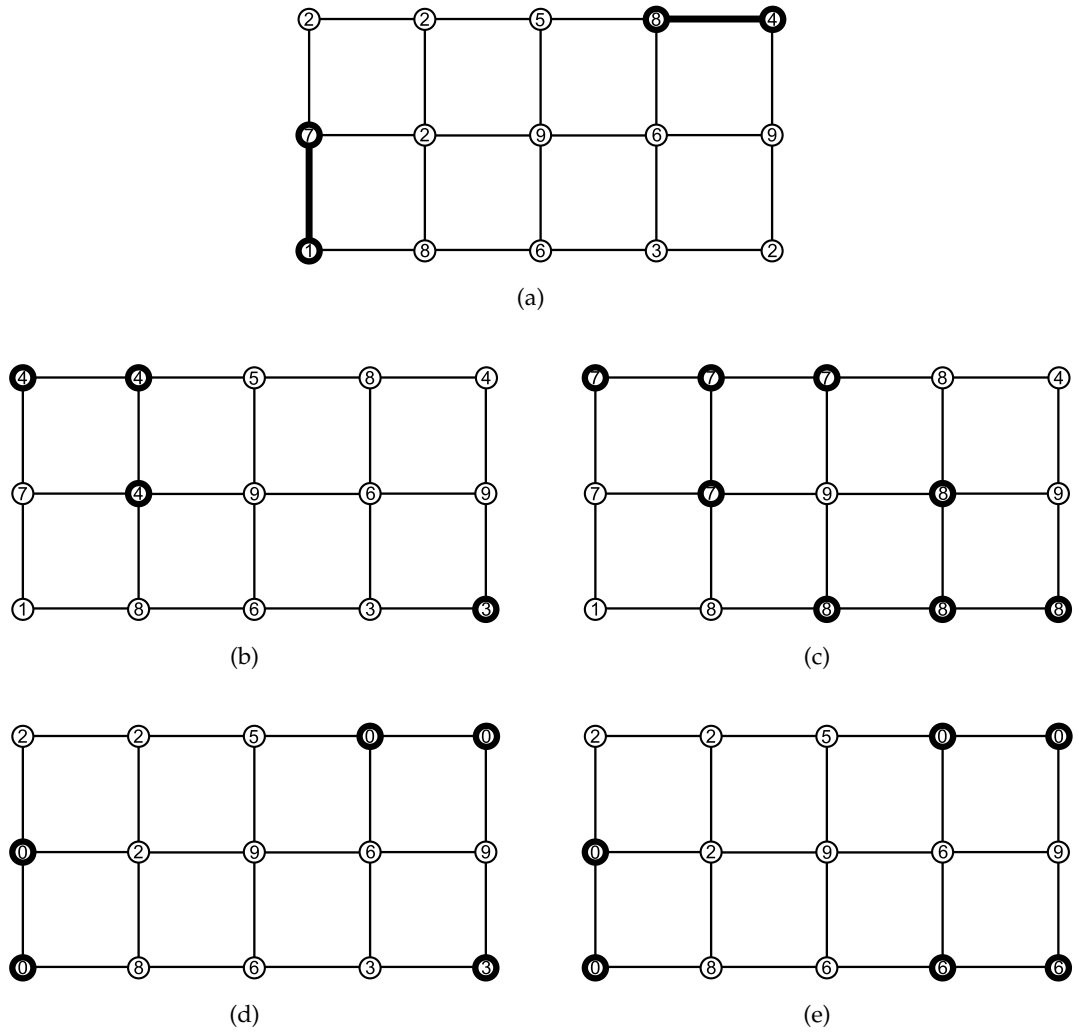


FIGURE 3.26 – Graphe G avec : (a) une application de poids P sur les sommets et (en gras) un sous-graphe M ; (b) une inondation de P contrainte par M et (en gras) les sommets de poids différents ; (c) une inondation maximale de P contrainte par M et (en gras) les sommets de poids différents ; (d) une inondation de P restreinte par M et (en gras) les sommets de poids différents ; (e) une inondation maximale de P restreinte par M et (en gras) les sommets de poids différents.

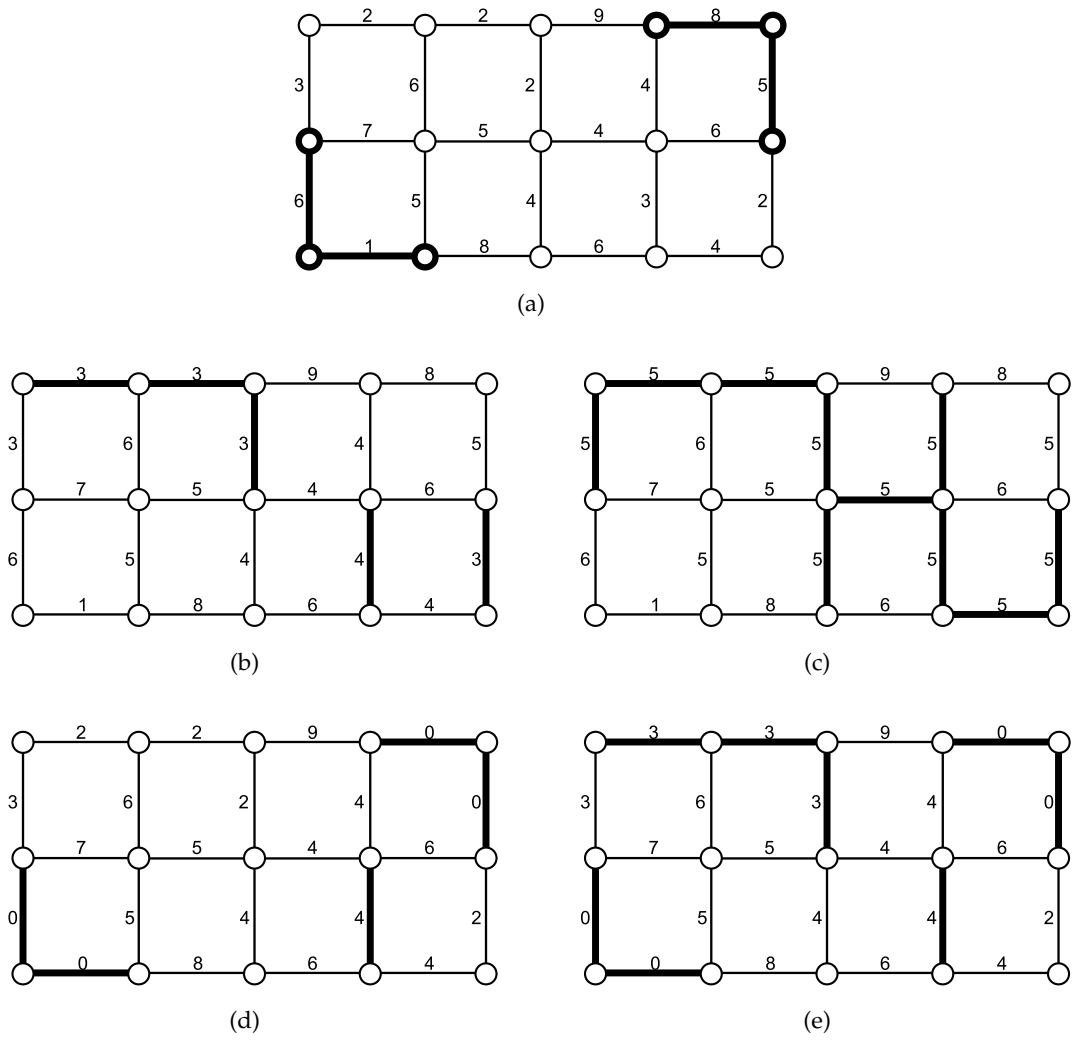


FIGURE 3.27 – Graphe G avec : (a) une application de poids P sur les arêtes et (en gras) un sous-graphe M ; (b) une inondation de P contrainte par M et (en gras) les arêtes de poids différents ; (c) une inondation maximale de P contrainte par M et (en gras) les arêtes de poids différents ; (d) une inondation de P restreinte par M et (en gras) les arêtes de poids différents ; (e) une inondation maximale de P restreinte par M et (en gras) les arêtes de poids différents.

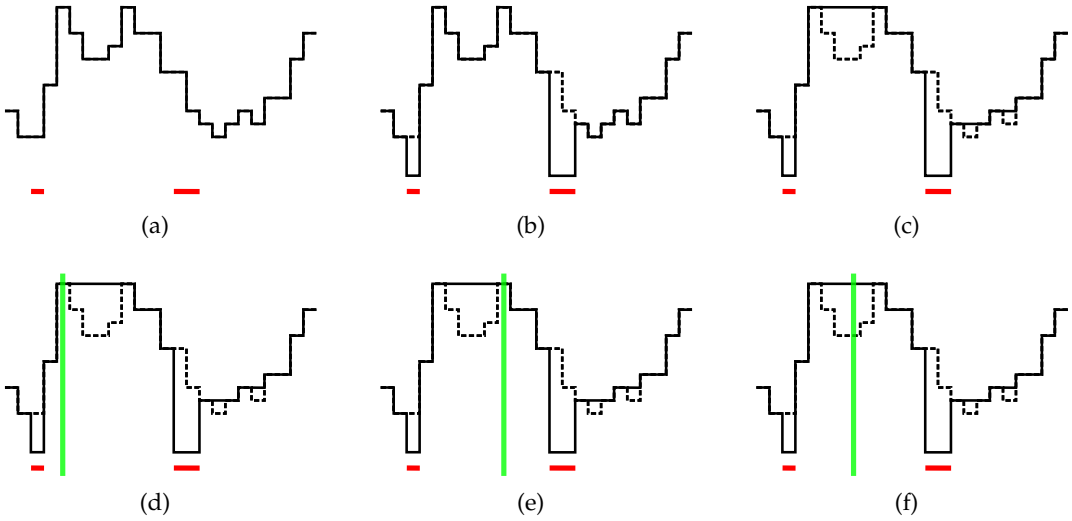


FIGURE 3.28 – Représentation 1D de : (a) une application de poids P avec un ensemble de marqueurs M (en rouge) ; (b) application de poids P' issue de P avec les marqueurs M comme poids les plus faibles (avec P en pointillés) ; (c) inondation maximale de P restreinte par M (avec P en pointillés) ; (d) à (f) trois différentes LPE (en vert) relatives à M.

L'inondation restreinte maximale, aussi appelée reconstruction morphologique (voir [205]), fut souvent effectuée sur les graphes à sommets valués puisque ceux-ci étaient le seul cadre servant aux LPE jusqu'à récemment. Son meilleur algorithme est quasi-linéaire (voir [205]).

Malheureusement, il arrive que le calcul d'une LPE relative à des marqueurs par cette méthode donne des résultats non désirés puisque la segmentation en résultant peut très bien passer au milieu d'un bassin versant de l'application de poids originale ou, pire, dans un de ses minima régionaux (voir les figures 3.28f, 3.29h et 3.29k).

3.5 Liens entre les diverses LPE

Bien que de définitions parfois très différentes, les paradigmes des différentes LPE présentées dans ce chapitre reposent tous sur le phénomène topographique éponyme, il est donc normal de retrouver certaines similitudes dans les résultats que l'on obtient. Nous verrons que c'est le cas dans les graphes de fusions parfaits pour les LPE sur graphes à sommets valués et, plus particulièrement encore, dans les graphes à arêtes valuées.

Dans [16] sont référencés certains liens entre LPE. Toutefois, cet article se concentre sur la comparaison de l'unicité ou non des résultats de LPE. Ainsi, il repose sur le principe de la LPE par zone d'affiliation (*tie-zone watershed* en anglais - [14, 17]) qui ne détermine pas de segmentation à proprement parler, mais la zone dans laquelle tout ensemble frontière est une LPE. La LPE par distance topographique est ainsi considérée comme un cas particulier de ce type de LPE.

Les liens présentés ci-après n'ont pas pour but une telle comparaison et offrent donc un point de vue complètement différent.

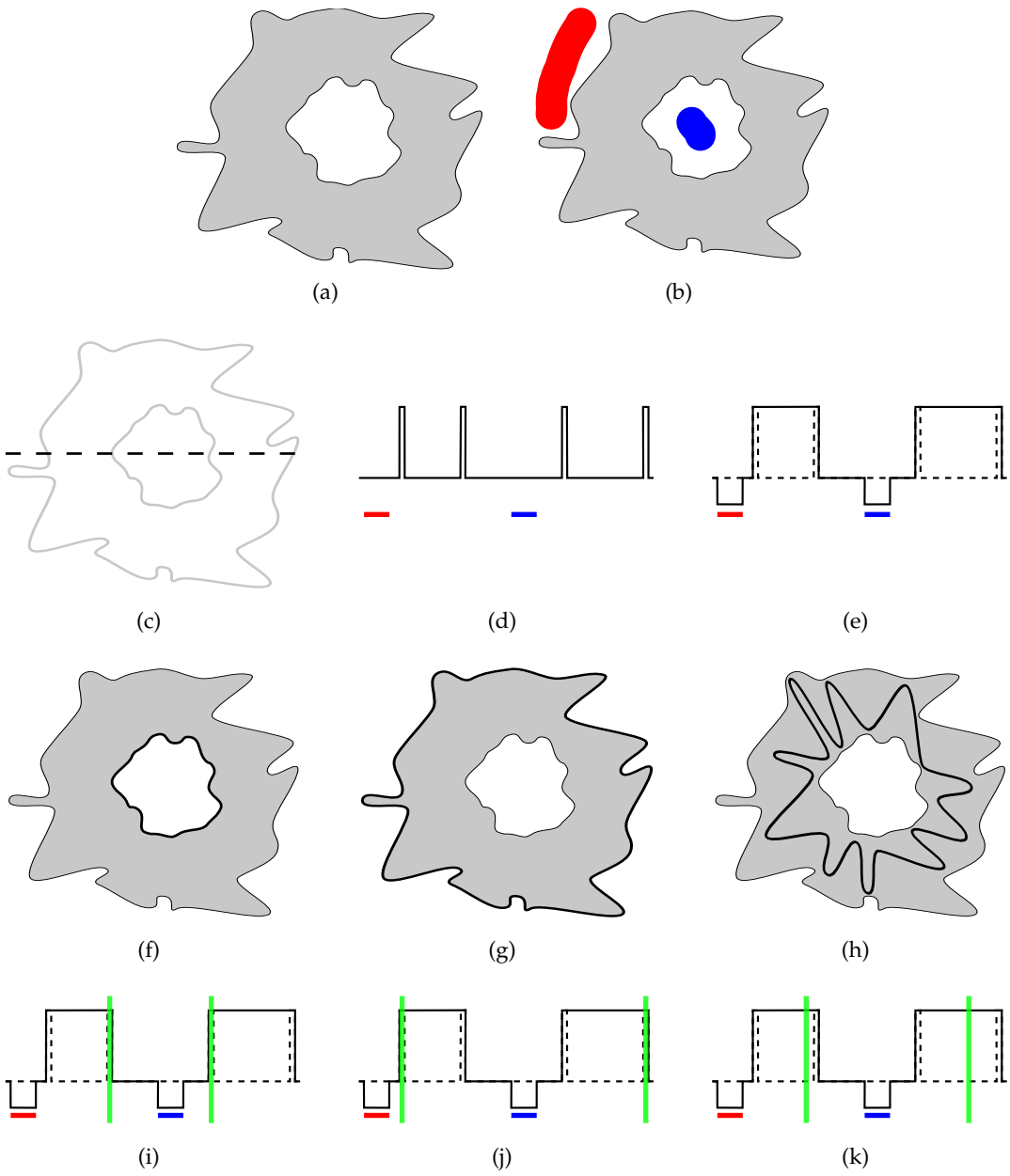


FIGURE 3.29 – LPE relative sur une image avec une application de poids de type gradient : (a) image d'entrée ; (b) marqueurs (deux ensembles : rouge et bleu) ; (c) gradient de l'image (a) ; (d) profile 1D du gradient (représenté par la ligne pointillée sur (c)) ; (e) profile 1D de l'inondation maximale restreinte aux marqueurs ; (f) à (h) LPE relatives aux marqueurs ; (i) à (k) profile 1D des LPE relatives respectives de (f) à (h).

3.5.1 Dans le cadre des graphes à sommets valués

Des liens entre les différentes LPE sur graphe à sommets valués présentées dans la section 3.2 ont déjà été mentionnés au fur et à mesure de leurs définitions. Nous les rappelons ci-dessous :

- tout ensemble frontière d'arêtes obtenu par amincissement d'une LPE par distance topographique est une LPE inter-sommets par immersion (théorème 3.14) ;
- LPE par immersion et LPE par érosion sont équivalentes (théorème 3.28).

3.5.2 Dans le cadre des graphes de fusion parfaite à sommets valués

Les LPE dans le cadre des graphes de fusion parfaite à sommets valués ont été étudiées dans [59, 60]. C'est d'ailleurs pour ce cadre qu'a été développée la LPE par érosion. En effet cette dernière possède des propriétés particulières dans ces graphes.

Il a déjà été vu dans la section 2.5.2 que, dans un graphe de fusion parfaite, tout ensemble frontière est fin. Toute LPE dont le résultat a pour propriété d'être un ensemble frontière (propriété P1) a donc également la propriété d'être fine (propriété P5) dans un graphe de fusion parfaite. Une LPE topologique, dans le cadre général, n'est pas nécessairement un ensemble frontière. Cependant, le cadre des graphes de fusion parfaite lui permet d'avoir la propriété suivante.

Propriété 3.57.
Si G est un graphe de fusion parfaite à sommets valués par l'application de poids P, alors toute LPE topologique de P est fine.

De plus, dans ce cadre, nous avons également les propriétés intéressantes suivantes qui nous permettent de mettre en évidence la relation entre érosion et ravinement.

Propriété 3.58.
Si G est un graphe de fusion parfaite à sommets valués par l'application de poids P, alors toute érosion de P est aussi un ravinement de P.

Il est à noter qu'un cloisonnement par érosion n'en devient pas pour autant un cloisonnement par ravinement. Toutefois, nous avons la propriété suivante.

Propriété 3.59.
Si G est un graphe de fusion parfaite à sommets valués par l'application de poids P, alors toute LPE par érosion de G pour P est aussi une LPE topologique de G pour P.

On déjà vu dans le théorème 3.28 qu'une LPE par immersion était équivalente à une LPE par érosion. Nous en déduisons donc que le lien de la LPE topologique avec la LPE par immersion est identique à celui qu'elle a avec la LPE par érosion.

Les graphes de fusion parfaite sont donc un cadre permettant principalement d'ajouter la propriété de finesse à certaines LPE présentées dans la section 3.2. Cependant, pour cela, il faut que l'application considérée dans laquelle intervient la LPE puisse se limiter à ce type de graphe.

3.5.3 Dans le cadre des graphes à arêtes valuées

Il a été mis en évidence dans la section 3.3 que les graphes à arêtes valuées étaient un cadre particulièrement propice à l'utilisation de LPE (avec, notamment, le théorème 3.32). Nous confirmerons cela dans cette section en montrant les liens qui existent avec les différentes LPE présentées dans la section 3.2 dont les définitions ont été directement adaptées pour pouvoir s'appliquer également dans le cadre des graphes à arêtes valuées.

3.5.3.1 Liens entre LPE par immersion, LPE par érosion et LPE d'arêtes

Les définitions de ravinement et d'érosion par bord présentées dans la section 3.3 ne sont pas sans rapport avec celles de ravinement et d'érosion présentées respectivement dans les sections 3.2.4 et 3.2.5 lorsqu'on les considère dans le cas de graphes à arêtes valuées comme le met en évidence la remarque 3.60.

Remarque 3.60.
Par définition, nous avons les deux inclusions suivantes :
- *un ravinement de bord est un ravinement ;*
- *une érosion de bord est une érosion.*

Néanmoins, il est à noter qu'une érosion minimale de bord n'est pas une érosion minimale. De plus, il n'y a pas d'inclusions équivalentes à celles-ci en ce qui concerne leurs différents cloisonnements respectifs.

Bien que les cloisonnements par bord ne peuvent entrer dans le cadre de la remarque 3.60, l'équivalence qui existe entre eux (présentée dans le théorème 3.45) se retrouve dans la propriété 3.61.

Propriété 3.61.
Soit une application de poids P sur les arêtes.
Si P' est un cloisonnement par bord de P, alors $Min(P')$ est une forêt couvrante relative à $Min(P)$ (autrement dit une extension couvrante minimale - voir la section 4.2).

La propriété 3.61 est ainsi à mettre en relation avec le fait que les ensembles frontières d'arêtes sont toujours fins. En conséquence de quoi les graphes induits par le complémentaire d'une LPE par immersion ou par érosion sont des extensions maximales des minima régionaux du graphe. On en déduit alors la propriété 3.62.

Propriété 3.62.
Soit une application de poids P sur les arêtes.
Si P' est un cloisonnement par érosion de P, alors $Min(P')$ est une extension (couvrante) maximale relative à $Min(P)$.

Les propriétés 3.61 et 3.62 mettent en évidence que les minima régionaux des cloisonnements par érosion minimale de bord et par érosion minimale sont des extensions couvrantes des minima de l'application de poids originale, minimale pour la première (autrement dit une forêt couvrante) et maximale pour la seconde. Mais la ressemblance ne s'arrête pas là comme le montre le théorème 3.63.

Théorème 3.63 ([58], théorème 21)**.**
Soient P une application de poids sur les arêtes et $L \subseteq A$ un ensemble d'arêtes.
Les trois propositions suivantes sont équivalentes :
- *L est une LPE d'arêtes de G pour P ;*
- *L est une LPE par immersion de G pour P ;*
- *L est une LPE par érosion de G pour P.*

3.5.3.2 Lien entre LPE par distance topographique et LPE d'arêtes

La remarque 3.33 de la section 3.3 a déjà mis en évidence le lien qui unit la LPE d'arêtes et la LPE par distance topographique dans les graphes à arêtes valuées. En effet, les minima d'un cloisonnement par bord forment une extension couvrante des bassins versants topographiques.

3.5.3.3 Lien entre la LPE topologique et LPE d'arêtes

Nous avons appris, dans la section 1.4.8.3, que les propriétés d'adjacence d'un graphe étaient conservées sur son graphe dérivé et réciproquement (remarque 1.21) et, dans la section 2.5.2, que tout graphe dérivé est un graphe de fusion parfaite (corollaire 2.31, en conséquence de quoi nous pouvons déduire des propriétés 3.58 et 3.59 les propriétés suivantes.

Propriété 3.64.
Dans un graphe à arêtes valuées, toute érosion est un ravinement.

La propriété suivante est une conséquence directe de cette dernière.

Propriété 3.65.
Dans un graphe à arêtes valuées, toute LPE par érosion est une LPE topologique.

Il est ensuite très facile de déduire du théorème 3.63 et de la propriété 3.65 la propriété ci-dessous qui fait ainsi le lien entre LPE d'arêtes et LPE topologique.

Propriété 3.66.
Dans un graphe à arêtes valuées, toute LPE d'arêtes est une LPE topologique.

3.5.3.4 Résumé des liens avec la LPE d'arêtes

Pour résumer la comparaison entre LPE d'arêtes et les quatre LPE décrites dans la section 3.2 qui a été faite ici, nous avons pu voir qu'une LPE d'arêtes est :
- équivalente à une LPE par immersion ;
- un amincissement des arêtes contenues dans une LPE par distance topographique ;
- une LPE topologique (mais la réciproque n'est en général pas vraie) ;
- équivalente à une LPE par érosion.

Cette section met donc en évidence les relations fortes qui existent, dans le cadre des graphes à arêtes valuées, entre toutes les différentes notions de LPE décrites jusqu'ici. Les propriétés qui semblaient manquer dans certaines de ces définitions ne sont donc pas nécessairement dues au principe inhérent à chaque méthode (basés sur des phénomènes naturels liés au cadre de la topographie), mais à leur cadre d'utilisation qui n'était pas adéquat. Les graphes à arêtes valuées, quant à elles, offrent un tel cadre.

3.6 Bilan des diverses LPE

Cette section vise à faire le bilan des différentes propriétés que possèdent les différentes versions de LPE présentées ici, d'exposer brièvement quelques variantes ayant pour but d'adapter la LPE à certains critères de la segmentation d'image.

3.6.1 Remarques générales sur les LPE et leur utilisation en segmentation d'images

Bien que logiquement basée sur les pixels à ses débuts, la LPE fut dès le départ utilisable dans le cadre des graphes à sommets valués. Ce n'est que plus récemment qu'elle fut adaptée dans le cadre des graphes à arêtes valuées.

Comme déjà mentionné dans la section 3.4, le calcul d'une LPE est souvent accompagné d'un prétraitement ayant pour but de réduire le nombre de minima régionaux de l'application de poids originale et ainsi limiter le nombre de régions obtenues au final. Ceci est particulièrement vrai lorsque l'on souhaite utiliser la LPE comme outil de segmentation d'image. En effet, on procède souvent à un lissage de sorte à éviter

la sur-segmentation. De plus, la segmentation ayant souvent pour but de détecter les contours d'objets, chaque région segmentée devant représenter au final un objet, une partie d'objet ou encore l'arrière-plan de l'image, la LPE est alors non pas appliquée sur l'image elle-même mais sur son gradient (morphologique ou autre). En effet, la norme d'un gradient donne de fortes valeurs, et donc de hautes crêtes si l'on se réfère au champ de la topographie, là où il y a de brusques variations de valeurs entre des pixels voisins de l'image originale, comme, par exemple, au bord d'un objet dont la couleur contraste avec le fond sur lequel il est présenté. Ce sont ensuite ces lignes de crêtes qui devraient être détectées en tant que LPE, déterminant alors les contours de l'objet.

Un phénomène inhérent au principe de LPE est le phénomène de fuite. Celui-ci est souvent considéré comme un défaut de ce type de méthode, notamment en segmentation d'images.En termes de graphe, il suffit donc qu'un seul élément d'un ensemble frontière de poids élevé soit de faible altitude, formant ainsi un col, pour que cet ensemble frontière ne soit pas une LPE. Ainsi, il suffit d'un "parasite" dans le résultat du gradient de l'image pour que la LPE passe complètement à côté du contour de l'image.

Nous verrons plus loin qu'il existe des méthodes visant à remédier à ceci.

3.6.2 Variantes de LPE

Il est à mentionner qu'il existe des variantes aux algorithmes de LPE présentés dans ce chapitre. Les modifications qui leur ont été apportées visent à adapter la LPE aux besoins d'une application en particulier ou à remédier au problème de fuite mentionné précédemment.

Ainsi, par exemple, dans [150, 203], la LPE par immersion est considérée comme se faisant dans un liquide visqueux et est donc appelée en conséquence LPE visqueuse. Cette viscosité empêche ainsi le liquide de passer dans de fines failles du relief, remédiant ainsi au problème de fuite. Cette méthode peut cependant empêcher, dans certains cas, la segmentation correcte de "détails fins".

Dans [158], un *a priori* de formes a été ajouté au calcul de la LPE. Ceci n'est bien entendu possible que pour des applications très spécifiques mais permet d'empêcher la sur-segmentation tout en évitant les phénomènes de fuite.

Dans [20], une contrainte sur la courbure du contour a été ajoutée au calcul de la LPE. Bien entendu ceci n'est faisable que si l'on connaît le type de courbure que possède le contour recherché.

Il est également à noter qu'une version de la LPE dans un espace continu à été proposé dans [155].

3.6.3 Quelle LPE privilégier ?

Le tableau 3.1 offre un récapitulatif des propriétés P1 à P5 que vérifie chacune des LPE présentées à travers ce chapitre dans le cadre de graphe quelconque à sommets ou arêtes valués correspondant à leur cadre d'origine respectif.

Dans les définitions de LPE décrites dans ce chapitre, l'ensemble d'éléments qui constitue la LPE est souvent du même type que celui sur lequel se fait la pondération. On remarque toutefois que, quel que soit le type d'élément sur lequel se fait la pondération, le fait d'avoir un ensemble frontière d'arêtes accroît le nombre de propriétés respectées par rapport à un ensemble frontière de sommets. Ceci confirme donc la conclusion du chapitre 2 qui privilégie l'utilisation d'ensembles frontières d'arêtes plutôt que de sommets.

Ceci se vérifie d'autant plus lorsque l'on travaille sur un graphe où ce sont les arêtes qui sont valuées et non les sommets. De plus, outre le respect de toutes les propriétés

Propriété	P1	P2	P3	P4	P5
LPE par immersion	OK	-	-	-	-
LPE inter-sommets par immersion	OK	*NA*	OK	*NA*	OK
LPE par distance topographique	-	*NA*	OK	*NA*	-
LPE topologique	-	-	-	OK	-
LPE par érosion	OK	-	-	-	-
LPE d'arêtes	OK	OK	OK	OK	OK

TABLE 3.1 – Propriétés des différentes LPE (*NA* = Non Applicable).

d'une LPE du domaine topographique qu'offre une LPE d'arêtes sur graphe à arêtes valuées, on a vu dans la section 3.5.3 que celle-ci avait des liens étroits avec les principes des LPE développées pour les graphes à sommets valués si on les adaptait pour le cas des arêtes valuées. Ce cadre nous permet donc d'"unifier" la plupart des définitions exposées ici comme l'ont déjà montré la remarque 3.33, le théorème 3.63 et la propriété 3.66.

En plus de cela, il est également à rappeler que les LPE d'arêtes sont les seules à avoir des algorithmes qui peuvent s'exécuter en temps linéaire.

Une difficulté qui peut apparaître lors de l'utilisation de LPE d'arêtes pour la segmentation d'images est le fait de transférer l'application de poids depuis les sommets sur les arêtes. En effet, il est naturel d'associer les valeurs d'intensité des pixels aux sommets du graphe d'adjacence (d'où le développement des premières LPE sur ce type de graphes) alors que nous venons de voir qu'il est préférable de travailler sur des arêtes valuées. Ce passage de l'application de poids P_S sur les sommets à l'application de poids P_A peut se faire de manière très simple en attribuant, par exemple, pour chaque arête $a = \{x, y\} \in A$ le poids $P_A(a) = \min\{P_S(x), P_S(y)\}$. Il est cependant à noter que cette opération peut entraîner la fusion ou la suppression de minima régionaux. Néanmoins, il est possible de considérer alors le calcul de la LPE relative à $Min(P_S)$ dans P_A. Bien entendu, tout dépend de l'application considérée.

Une autre alternative se prête particulièrement bien au cas où l'on souhaite procéder au calcul d'une LPE sur le gradient d'une image. En effet, il suffit alors de prendre l'application de poids sur les arêtes P_A à partir de l'application de poids P_S en prenant comme poids pour chaque arête $a = \{x, y\} \in A$ la norme de la différence entre les poids de ses sommets : $P_A(a) = \|P_S(x) - P_S(y)\|$. L'application de poids P_A aura donc une valeur dans $\mathbb{R}^+$ qui sera élevée si les valeurs dans P_S de ses sommets extrémités diffèrent beaucoup ou au contraire nulle si elles sont identiques. Les poids ainsi attribués font donc office de gradient et permettent de calculer une LPE directement. De plus, contrairement au cas classique, cela permet d'utiliser des poids de nature quelconque pour l'application P_S du moment que l'on utilise une norme adaptée pour créer P_A. On peut ainsi très facilement calculer une LPE sur une image couleur (de composantes RGB par exemple) alors que pour une LPE appliquée sur le graphe à sommets valués correspondant il aurait fallu d'abord faire une conversion en niveaux de gris.

La représentation d'une LPE en tant qu'ensemble frontière de sommets peut-être appréciable dans certaines applications afin de visualiser sur les sommets où se trouve la frontière entre les régions. Nous avons toutefois vu dans la section 2.6 qu'il n'est pas toujours aisé de passer d'un ensemble frontière d'arêtes à un ensemble frontière de sommets.

L'ensemble des résultats présentés dans ce chapitre nous permet donc de préconiser

l'utilisation de LPE d'arêtes. Ainsi, dans la suite de cette thèse, sauf mention explicite, quand il sera question de LPE, c'est à la définition de LPE d'arêtes qu'il sera fait référence.

Nous allons voir, dans le chapitre 4, que le cadre offert par cette définition permet d'établir des liens entre LPE et certaines forêts couvrantes relatives déjà mentionnées dans ce chapitre et que nous définirons dans la section 4.2, lui conférant ainsi de nouveaux éléments en sa faveur.

Arbres et forêts

4

Après avoir abordé la segmentation de graphe dans le chapitre 2 et différentes versions de ligne de partage des eaux formant autant de paradigmes de segmentation de graphe dans le chapitre 3, nous présentons dans ce chapitre les concepts d'arbres et de forêts dans les graphes qui, avec certains critères, permettent d'obtenir une segmentation de graphe, en l'occurrence une coupe, et donc une partition de ses sommets.

Dans le domaine des graphes, les arbres et les forêts sont des graphes non-orientés particuliers qui peuvent être définis de plusieurs façons. Toutes sont équivalentes mais certaines permettent de mettre en évidence des propriétés plus aisément que d'autres.

Nous introduisons ensuite deux sortes de forêts optimales sur graphes non-orientés à arêtes valuées, les forêts couvrantes de poids extremum et les forêts couvrantes de chemins de moindre altitude, qui permettent donc de segmenter un graphe selon un critère spécifique.

Nous rappelons les comparaisons déjà faites entre forêts couvrantes de poids minimum et de chemins de moindre altitude avant de présenter de nouveaux liens existant entres ces forêts couvrantes et les LPE d'arêtes (introduites dans la section 3.3.

Les notions traitées dans ce chapitre concernent uniquement le cas de graphes non-orientés. C'est pourquoi, dans la suite de ce chapitre, nous considérons le graphe non-orienté $G = (S, A)$. De plus, la pondération du graphe se fait systématiquement sur les arêtes avec l'application de poids $P : A \to \mathbb{R}$.

Sommaire du chapitre

Figures du chapitre

4.1 Définitions "classiques"

Soit G un graphe non-orienté. On dit que G est un **arbre** si l'une des affirmations suivantes (qui sont équivalentes) est vérifiée :

1. G est connexe et acyclique ;
2. G est connexe et $\forall a \in A$, $G' = (S, A \setminus \{a\})$ n'est pas connexe ;
3. G est acyclique et $\forall a \in S^2 \setminus A$, $G' = (S, A \cup \{a\})$ n'est pas acyclique ;
4. G est connexe et $|A| = |S| - 1$;
5. G est acyclique et $|A| = |S| - 1$;
6. il existe une unique chaîne élémentaire reliant toute paire de sommets de G.

Une forêt est un graphe dont chaque composante connexe est un arbre. Ainsi, on dit que G est une **forêt** si l'une des affirmations suivantes (qui sont équivalentes) est vérifiée :

1. G est acyclique ;
2. $\forall a \in A$, $G' = (S, A \setminus \{a\})$ a une composante connexe de plus que G ($|\mathcal{C}(G')| = |\mathcal{C}(G)| + 1$) ;
3. $|A| = |S| - |\mathcal{C}(G)|$;
4. il existe une unique chaîne élémentaire reliant toute paire de sommets d'une même composante connexe de G.

Remarque 4.1.
Il est à noter que si l'on part d'un graphe G qui est une forêt pour construire un graphe G' en y rajoutant un sommet et $|\mathcal{C}(G)|$ arêtes reliant ce sommet à chacune des composantes connexes de G, alors G' est un arbre.

4.2 Définitions relatives à un sous-graphe

Soit G un graphe non-orienté et soient F et M deux sous-graphes de G. On dit que F est une **forêt relative** à M (sur G) si :

– F est une extension de M, et
– l'une des affirmations suivantes (qui sont équivalentes) est vérifiée :

1. $F \setminus M$ est acyclique ;
2. $\forall a \in A(F \setminus M)$, $F' = (S(F \setminus M), A(F \setminus M) \setminus \{a\})$ a une composante connexe de plus que $F \setminus M$ ($|\mathcal{C}(F')| = |\mathcal{C}(F \setminus M)| + 1$) ;
3. $|A(F \setminus M)| = |S(F \setminus M)| - |\mathcal{C}(F \setminus M)|$;
4. il existe une unique chaîne élémentaire reliant toute paire de sommets d'une même composante connexe de $F \setminus M$.
5. il est impossible de supprimer une arête de $F \setminus M$ tout en gardant la propriété d'extension ;
6. pour toute extension $X \subseteq F$ de M, $V(X) = V(F) \Rightarrow X = F$.

Le sous-graphe M est souvent appelé **marqueur** pour les applications liées à la segmentation.

On remarque aisément que si G est connexe et $M = (V_M, \emptyset)$ où $V_M \subseteq V$ (i.e. M est un sous-graphe de G sans aucune arête, donc composé de sommets isolés), alors la notion de forêt relative à M correspond exactement à la notion traditionnelle de forêt présentée dans la section 4.1. De plus, si $|V_M| = 1$, on retrouve alors la notion usuelle d'arbre.

Ainsi, il est intéressant de noter que les propriétés classiques des arbres et forêts peuvent s'appliquer aux arbres et forêts relatifs à des marqueurs puisque, en remplaçant

chaque composante connexe du marqueur par un sommet isolé, on retrouve la notion usuelle d'arbre ou de forêt.

Soit G un graphe non-orienté et soient F et M deux sous-graphes de G. On dit que F est une **forêt couvrante relative** à M (sur G) si :
- F est une forêt relative à M, et
- $V(F) = V$.

Remarque 4.2.
On peut remarquer que F est une forêt couvrante relative à M (sur G) s'il existe une extension couvrante X relative à M sur G telle que F soit obtenue en supprimant autant d'arêtes de X que possible tout en préservant la propriété d'extension couvrante.

Remarque 4.3.
Il est à noter que si l'on part d'un graphe G qui est une forêt relative à M pour construire un graphe G′ en réduisant chaque composante connexe de M en un seul sommet ayant les mêmes adjacences que la composante connexe correspondante, on obtient une forêt "classique" qui peut à son tour être ramenée à un arbre, comme exposé dans la remarque 4.1.

Remarque 4.4.
Par définition, une forêt couvrante relative est une ***extension couvrante minimale relative****.*

Des exemples de ces définitions se trouvent dans la figure 4.1.

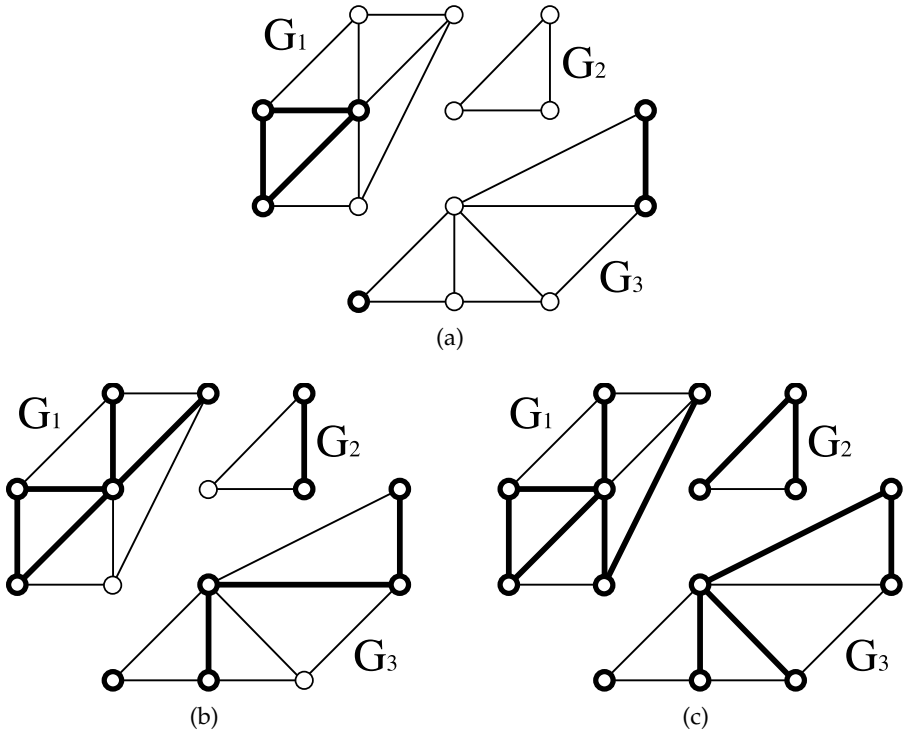


FIGURE 4.1 – Graphe G, composé de trois composantes connexes (G_1, G_2 et G_3), avec (en gras) : (a) un sous-graphe M ; (b) une forêt relative à M ; (c) une forêt couvrante relative à M.

4.3 Forêt couvrante de poids extremum

Soit G un graphe non-orienté avec une application de poids P sur les arêtes et soient F et M deux sous-graphes de G.

On dit que F est une **forêt couvrante de poids minimum** (**FCMin**), respectivement **de poids maximum** (**FCMax**), **relative** à M (pour P sur G) si :
- F est une forêt couvrante relative à M, et
- le poids de F est minimum (i.e. pour toute forêt couvrante F' relative à M, $P(F) \leq P(F')$), respectivement maximum (i.e. pour toute forêt couvrante F' relative à M, $P(F) \geq P(F')$).

Afin d'alléger l'écriture, on appelle **coupe par FCMin** l'unique coupe induite par une FCMin. Respectivement, on appelle **coupe par FCMax** l'unique coupe induite par une FCMax.

Des exemples de ces définitions se trouvent dans la figure 4.2.

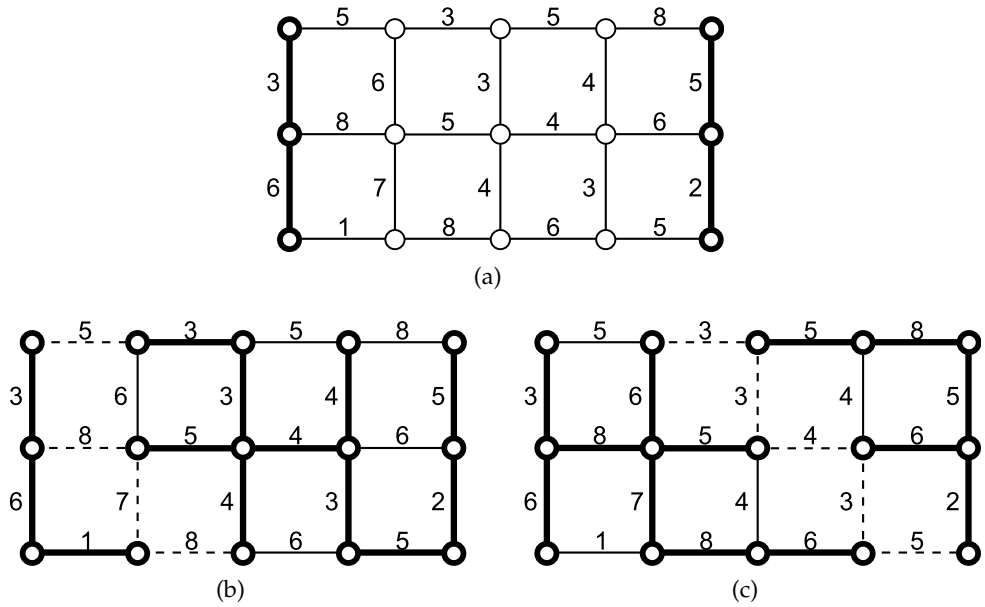


FIGURE 4.2 – Graphe G à arêtes valuées par P avec : (a) en gras, un sous-graphe M ; (b) en gras, une FCMin relative à M et, en pointillés, la coupe induite par cette FCMin ; (c) en gras, une FCMax relative à M et, en pointillés, la coupe induite par cette FCMax.

Remarque 4.5.
Soient $f : \mathbb{R}^+ \to \mathbb{R}^+$ une fonction strictement croissante et $g : \mathbb{R}^+ \to \mathbb{R}^+$ une fonction strictement décroissante.
De par des résultats classiques sur les forêts couvrantes de poids extremum (voir [128], chapitre 6.1), on sait que les trois énoncés suivants sont équivalents :
- *F est une FCMin relative à M pour P ;*
- *F est une FCMin relative à M pour $(f \circ P)$;*
- *F est une FCMax relative à M pour $(g \circ P)$.*

Nous avons donc également les trois équivalences suivantes :
- *F est une FCMax relative à M pour P ;*
- *F est une FCMax relative à M pour $(f \circ P)$;*
- *F est une FCMin relative à M pour $(g \circ P)$.*

D'après la remarque 4.5, une fonction strictement décroissante permet de passer d'une FCMin à une FCMax et réciproquement. C'est dans cette optique que, dans la suite de cette section, nous ne traiterons que le cas de la recherche de FCMin, le cas de la recherche d'une FCMax étant traité par la même occasion, à une fonction strictement décroissante près.

La recherche d'un arbre de poids minimum est un problème typique d'optimisation combinatoire étudié depuis de nombreuses années (voir [129, 157, 171]). Il fut, entre autres, utilisé en analyse d'image [212]. En se basant sur la construction présentée dans [144], similaire à la construction des remarques 4.1 et 4.3 (auquel cas les poids des arêtes rajoutées sont inférieurs au poids le plus faible du graphe original), on déduit que la recherche d'une FCMin est équivalente à la recherche d'un arbre de poids minimum. En conséquence de quoi tout algorithme de recherche d'arbre de poids minimum peut être aisément adapté à la recherche d'une FCMin relative à des marqueurs. Un état de l'art d'algorithmes résolvant ce problème peut être trouvé dans [53, 101, 128]). Plus récemment, un algorithme quasi-linéaire (au sens de l'*union-find* de Robert Endre Tarjan [198]) fut proposé par Bernard Chazelle dans [48] et qui est, à l'heure actuelle, le meilleur algorithme connu avec une complexité en $O(|A|\alpha(|A|,|S|)$ où α est l'inverse de la fonction de Ackermann. Cette dernière, établie par Wilhelm Ackermann en 1928 [1], prend en entrée des entiers positifs et renvoie un entier supérieur ou égal à 1. Sa particularité est d'avoir une croissance extrêmement rapide. Son inverse, notée α, croît donc extrêmement lentement et est dite comme inférieure à 5 pour toute valeur d'entrée "concevable". Ainsi, pour toute application pratique, α peut être considérée comme une constante. En conséquence de quoi, l'algorithme de Bernard Chazelle a une complexité extrêmement proche du cas linéaire.

4.4 Forêt couvrante de chemins de moindre altitude

Soit G un graphe non-orienté avec une application de poids P sur les arêtes et soient F et M deux sous-graphes de G.

On dit que F est une **forêt de chemins de moindre altitude** (**FCMA** - *shortest path forest* en anglais) **relative** à M (pour P sur G) si :

- F est une forêt relative à M, et
- pour tout sommet $s \in S(F)$ il existe une chaîne π de s à M dans F telle que l'altitude de π soit égale à l'altitude de connexion entre s et M (i.e. $H(\pi) = H(x, M)$ - voir la sous-section 1.4.9.2).

On dit que F est une **forêt couvrante de chemins de moindre altitude** (**FCCMA** - *shortest path spanning forest* en anglais) **relative** à M (pour P sur G) si :

- F est une forêt couvrante relative à M, et
- F est une forêt de chemins de moindre altitude.

Cette fois encore, afin d'alléger l'écriture, On appelle **coupe par FCCMA** l'unique coupe induite par une FCCMA.

Des exemples de ces définitions se trouvent dans la figure 4.3.

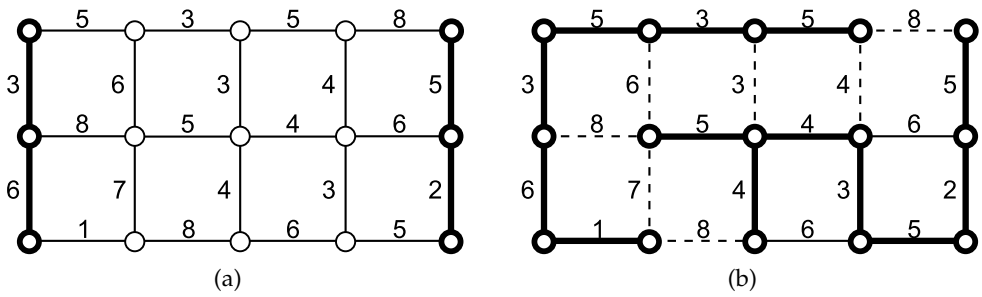


Figure 4.3 – Graphe G à arêtes valuées par P, avec : (a) en gras, un sous-graphe M ; (b) en gras, une FCCMA relative à M et, en pointillés, la coupe induite par cette FCCMA.

Il est à noter que les transformées image-forêt (voir [83]) et la segmentation d'image par connexité floue (voir [15, 181, 201, 202]) sont des cas particuliers des forêts couvrantes de chemins de moindre altitude.

4.5 Lien entre forêt couvrante de chemins de moindre altitude et forêt couvrante de poids minimum

Dans cette section, nous rappelons tout d'abord une propriété des forêts couvrantes de poids minimum vis-à-vis des altitudes de connexion, puis deux théorèmes liant forêt couvrante de poids minimum et forêt couvrante de chemins de moindre altitude.

Soit G un graphe non-orienté avec une application de poids P sur les arêtes et soient F et M deux sous-graphes de G.

Propriété 4.6 ([58], théorème 29)**.**
Si F est une FCMin relative à M, alors, pour tout couple de composantes connexes M_X et M_Y de M nous avons $H(M_X, M_Y) = H(F_X, F_Y)$ où F_X et F_Y sont les composantes connexes de F qui incluent respectivement M_X et M_Y ($M_X \subseteq F_X$ et $M_Y \subseteq F_Y$).

Les théorèmes qui suivent sont des conséquences de la propriété 4.6.

Théorème 4.7 ([58], propriété 30)**.**
Si F est une FCMin relative à M, alors F est une FCCMA relative à M.

On déduit aisément du théorème 4.7 le corollaire suivant.

Corollaire 4.8.
Toute coupe par FCMin est une coupe par FCCMA.

Remarque 4.9.
Il est à noter que la réciproque du théorème 4.7 n'est, en général, pas vraie.

Une illustration de cette remarque se trouve dans la figure 4.4.

Dans le cas particulier où le sous-graphe auquel sont relatives les forêts couvrantes est $Min(P)$, nous obtenons le théorème suivant.

Théorème 4.10 ([58], propriété 31)**.**
Le sous-graphe F est une FCMin relative à $Min(P)$ si et seulement si F est une FCCMA relative à $Min(P)$.

On déduit aisément du théorème 4.7 le corollaire suivant.

Corollaire 4.11.
Une coupe relative à $Min(P)$ est une coupe par FCMin si et seulement si elle est une coupe par FCCMA.

Maintenant que les liens entre les différentes forêts couvrantes ont mis en évidence, nous montrerons dans les sections suivantes (4.6 et 4.7) ainsi que dans le chapitre 5 que ces forêts couvrantes partagent également des liens avec d'autres critères de segmentation de graphe.

4.6 Forêt couvrante de chemins de moindre altitude et ligne de partage des eaux

Dans cette section se trouvent les résultats mettant en évidence les liens existant entre les forêts couvrantes de chemins de moindre altitude et les LPE, relatives à $Min(P)$ dans un premier temps, puis, relatives à un sous-graphe quelconque dans un second.

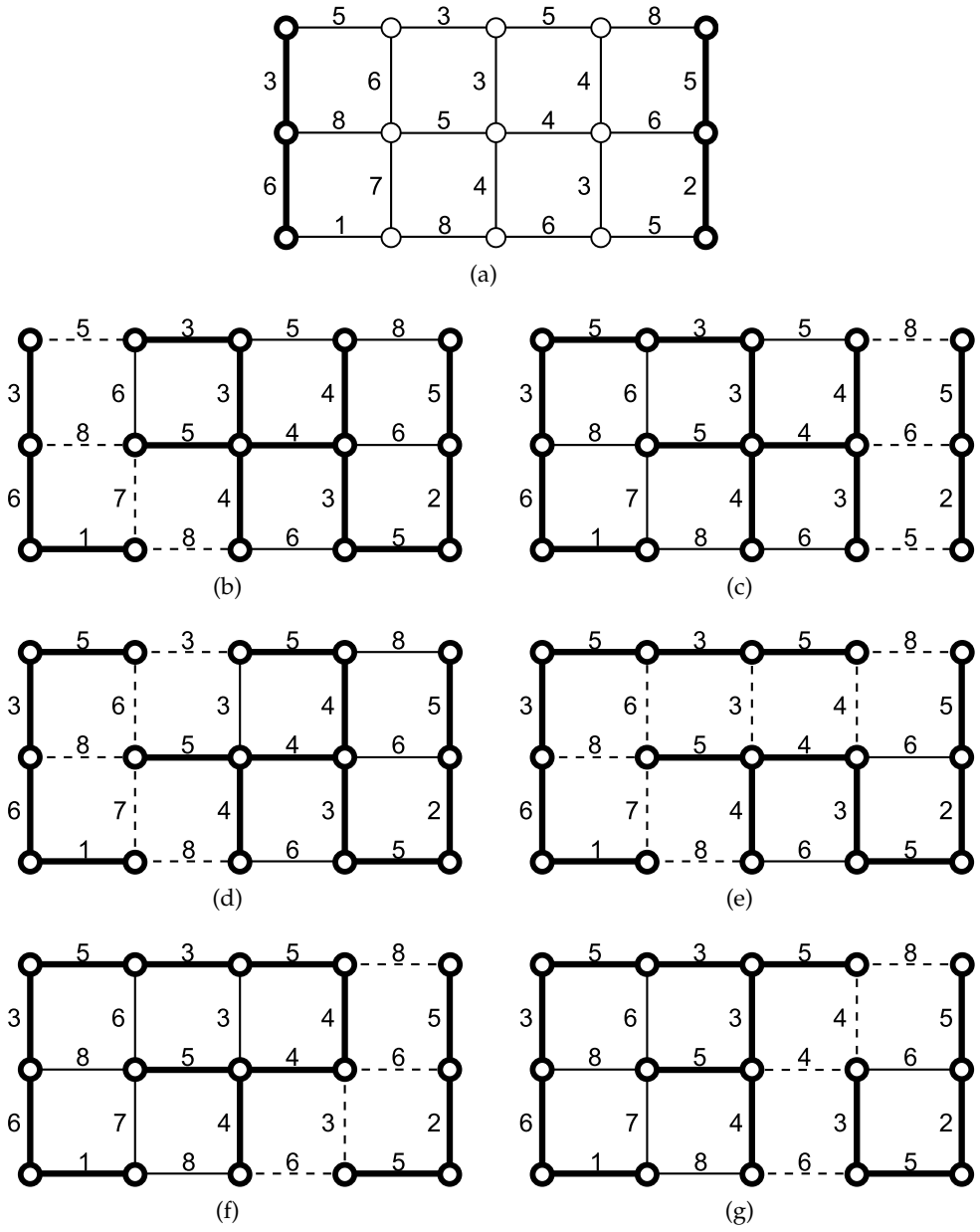


FIGURE 4.4 – Graphe G à arêtes valuées par P avec (en gras) : (a) un sous-graphe M ; (b,c) deux FCMin relative à M qui sont donc des FCCMA relatives à M ; (d) à (g) quatre FCCMA relatives à M qui ne sont pas des FCMin relatives à M. Les arêtes des coupes induites par les forêts couvrantes sont représentées en pointillés.

4.6.1 FCCMA et LPE relatives à $Min(P)$

On déduit aisément des théorèmes 4.10 et 4.16 le théorème suivant.

Théorème 4.12.
Soit un ensemble d'arêtes $C \subseteq A$.
L'ensemble d'arêtes C est une coupe par FCCMA relative à $Min(P)$ (pour P) si et seulement si C est une LPE (pour P).

Une illustration de ce théorème se trouve dans la figure 4.5.

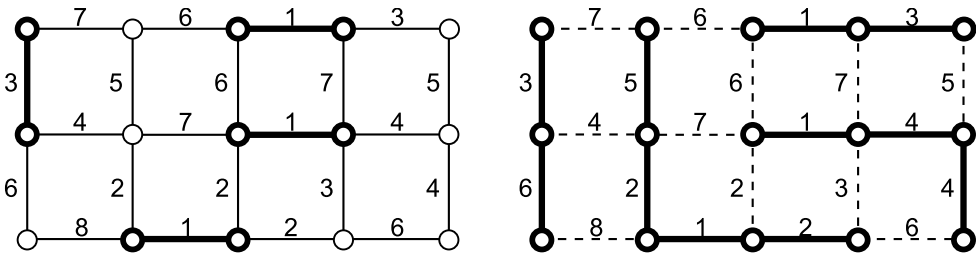


FIGURE 4.5 – Graphe G à arêtes valuées par l'application de poids P avec : (a) en gras, un sous-graphe $M = Min(P)$; (b) en gras, une FCCMA relative à M pour P et, en pointillés, sa coupe induite C. On peut remarquer, d'après la définition 3.30, que C est effectivement une LPE.

4.6.2 FCCMA et LPE relatives à un sous-graphe quelconque

Nous présentons maintenant les théorèmes qui nous ont permis de déterminer le lien existant entre LPE et FCCMA relatives à un sous-graphe quelconque.

Théorème 4.13.
Soit M un sous-graphe de G. Soient une application de poids P et E l'ensemble d'éléments de G sur lequel s'effectue la pondération.
Soit une inondation P' de P restreinte par M ($P' \in \tilde{\mathcal{F}}(P, M)$).
Pour tout sommet $s \in S$ et pour toute chaîne π de s à M, l'altitude de π pour P est égale à l'altitude de π pour P'.

Preuve du théorème 4.13.
Soit M_π la composante connexe de M qui contient l'extrémité de π.
Il est évident que, pour tout sommet $s \in S$ et pour toute chaîne π de s à M, $H_P(\pi) = H_{P'}(\pi)$.
Supposons maintenant qu'il existe une chaîne π telle que $H_P(\pi) \neq H_{P''}(\pi)$, autrement dit, que $H_{P'}\pi) \neq H_{P''}(\pi)$. Si $H_{P'}(\pi) > H_{P''}(\pi)$, cela signifie que le poids d'au moins une arête de G a été abaissé en passant de P' à P''. Or, d'après la remarque 3.48, cela est impossible. Dans ce cas, nous avons alors forcément $H_{P'}(\pi) < H_{P''}(\pi)$. Considérons un réel h tel que $H_{P'}(\pi) < h < H_{P''}(\pi)$. Par construction, nous avons alors la composante connexe de $G^h_{P''}$ contenant M_π qui est strictement incluse dans la composante connexe de $G^h_{P'}$ contenant M_π. Or, par définition d'une inondation, cela est impossible. En conséquence de quoi P'' ne peut être une inondation, ce qui contredit l'hypothèse.
Nous avons donc bien, pour tout sommet $s \in S$ et pour toute chaîne π de s à M, $H_P(\pi) = H_{P''}(\pi)$. □

On déduit aisément du théorème 4.13 le théorème suivante.

Théorème 4.14.
Soit M un sous-graphe de G. Soit P' une inondation de P ($P' \in \mathcal{F}(P)$).
Le sous-graphe F de G est une FCCMA relative à M pour P si et seulement si F est une FCCMA relative à M pour P''.

Une illustration de ce théorème se trouve dans la figure 4.6.

Le théorème 4.14 nous permet de trouver le théorème suivant.

Théorème 4.15.
Soit M un sous-graphe de G et $C \subseteq A$ un sous-ensemble d'arêtes.
L'ensemble d'arêtes C est une coupe par FCCMA relative à M pour P si et seulement si C est une LPE relative à M pour P.

Preuve du théorème 4.15.
Notons P' l'inondation maximale de P restreinte par M.
D'après la définition 3.56 et le théorème 4.12 le théorème 4.10, nous savons que toute coupe C relative à M est une coupe par FCCMA relative à M pour P' si et seulement si C est une LPE relative à M. On déduit alors du théorème 4.14 que C est une coupe par FCCMA relative à M pour P si et seulement si C est une LPE relative à M. □

Une illustration de ce théorème se trouve dans la figure 4.7.

Cette section a ainsi mis en évidence que la recherche d'une LPE, relative ou non, peut toujours se faire par la recherche d'une coupe induite par FCCMA sur l'application de poids d'origine. Dans le cas d'une LPE relative, la reconstruction morphologique n'est alors plus nécessaire.

4.7 Forêt couvrante de poids minimum et ligne de partage des eaux

Dans cette section se trouvent les résultats mettant en évidence les liens et implications existant entre les forêts couvrantes de poids minimum et les LPE, relatives à $Min(P)$ dans un premier temps, puis, relatives à un sous-graphe quelconque dans un second.

4.7.1 FCMin et LPE relatives à $Min(P)$

Nous rappelons dans le théorème 4.16 le lien, prouvé dans [7, 58], qui unit FCMin et LPE relatives aux minima régionaux d'une application de poids.

Théorème 4.16.
Soit un ensemble d'arêtes $C \subseteq A$.
L'ensemble d'arêtes C est une coupe par FCMin relative à $Min(P)$ (pour P) si et seulement si C est une LPE (pour P).

Une illustration de ce théorème se trouve dans la figure 4.8.

Il est à noter que le théorème 4.16 découle de la propriété 3.61 selon laquelle les minima régionaux des cloisonnements par bords, dont la coupe induite est une LPE d'après le théorème 3.46, sont des forêts couvrantes relatives à $Min(P)$.

4.7.2 FCMin et LPE relatives à un sous-graphe quelconque

Nous présentons maintenant les théorèmes qui nous ont permis de déterminer le lien existant entre FCMin et LPE relatives à un sous-graphe quelconque.

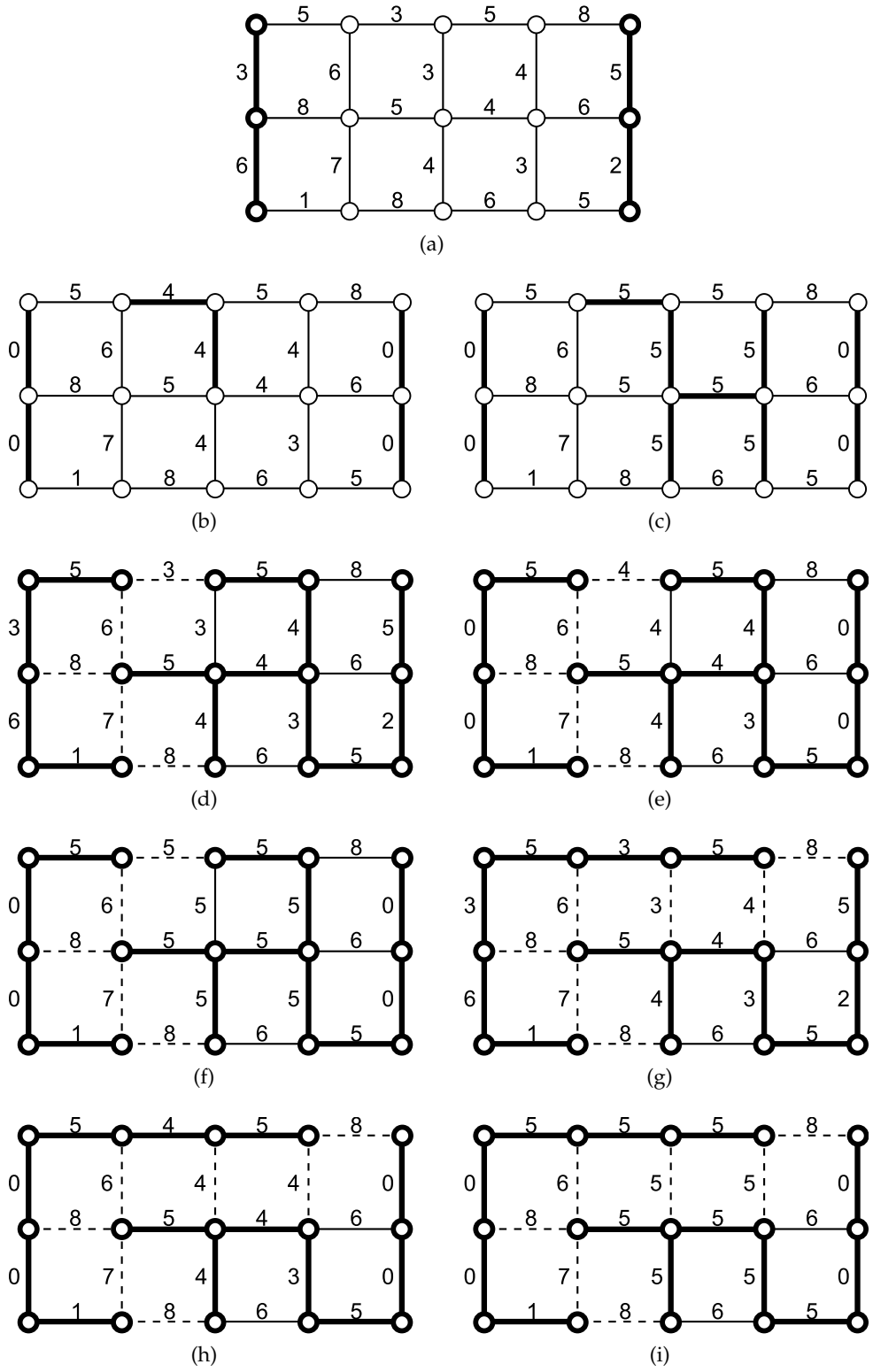


FIGURE 4.6 – Graphe G avec : (a) une application de poids P et, en gras, un sous-graphe M ; (b) inondation P' de P restreinte par M (avec les arêtes de poids différents en gras) ; (c) inondation maximale P'' de P restreinte par M (avec les arêtes de poids différents en gras) ; (d) à (f) FCCMA F_1 relative à M sur G pour, respectivement, P, P' et P'' ; (g) à (i) FCCMA F_2 relative à M sur G pour, respectivement, P, P' et P''. Les arêtes des coupes induites par les FCCMA sont représentées en pointillés.

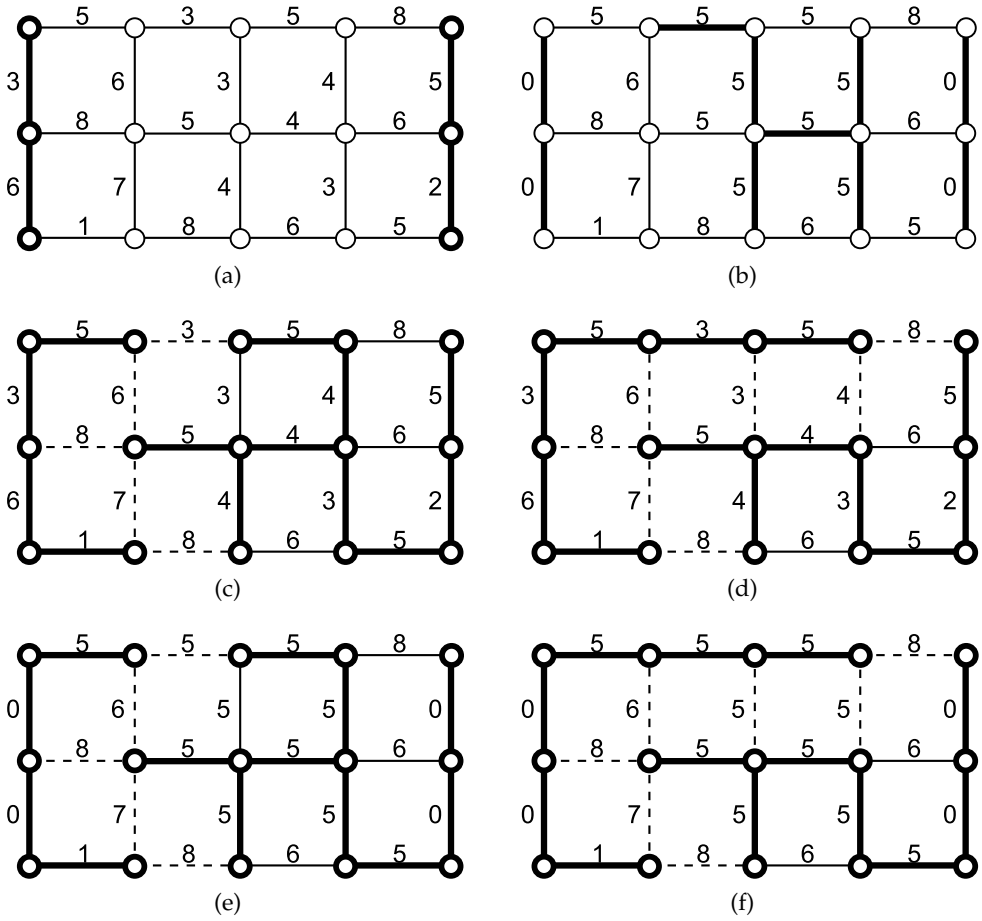


FIGURE 4.7 – Graphe G avec : (a) une application de poids P et, en gras, un sous-graphe M ; (b) inondation maximale P' de P restreinte par M (avec les arêtes de poids différents en gras) ; (c,d) respectivement F_1 et F_2 qui sont deux FCCMA relatives à M sur G pour P ; (f,g) respectivement F_1 et F_2 qui sont deux FCCMA relatives à M sur G pour P' également. Les arêtes des coupes induites par les FCCMA, qui sont également des LPE relatives à M, sont représentées en pointillés.

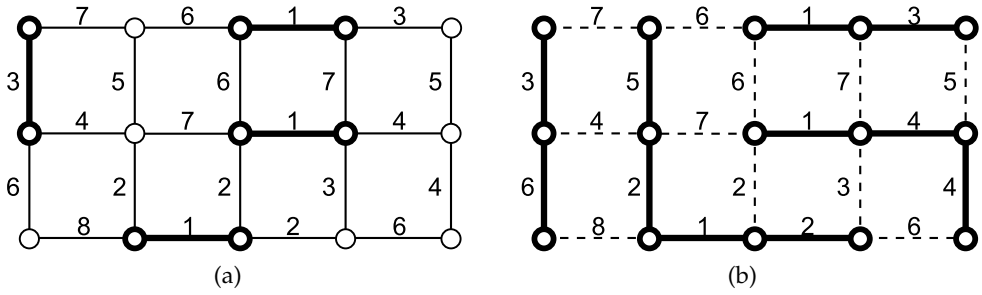


FIGURE 4.8 – Graphe G à arêtes valuées par l'application de poids P avec : (a) en gras, un sous-graphe $M = Min(P)$; (b) en gras, une FCMin relative à M pour P et, en pointillés, sa coupe induite C. On peut remarquer, d'après la définition 3.30, que C est effectivement une LPE.

Théorème 4.17.
Soit M un sous-graphe de G et P' une inondation de P ($P' \in \mathcal{F}(P)$).
Si F est une FCMin relative à M sur G pour P, alors F est une FCMin relative à M sur G pour P'.

La preuve de ce théorème se trouve ci-après, en fin de section.

Une illustration de ce théorème se trouve dans la figure 4.9.

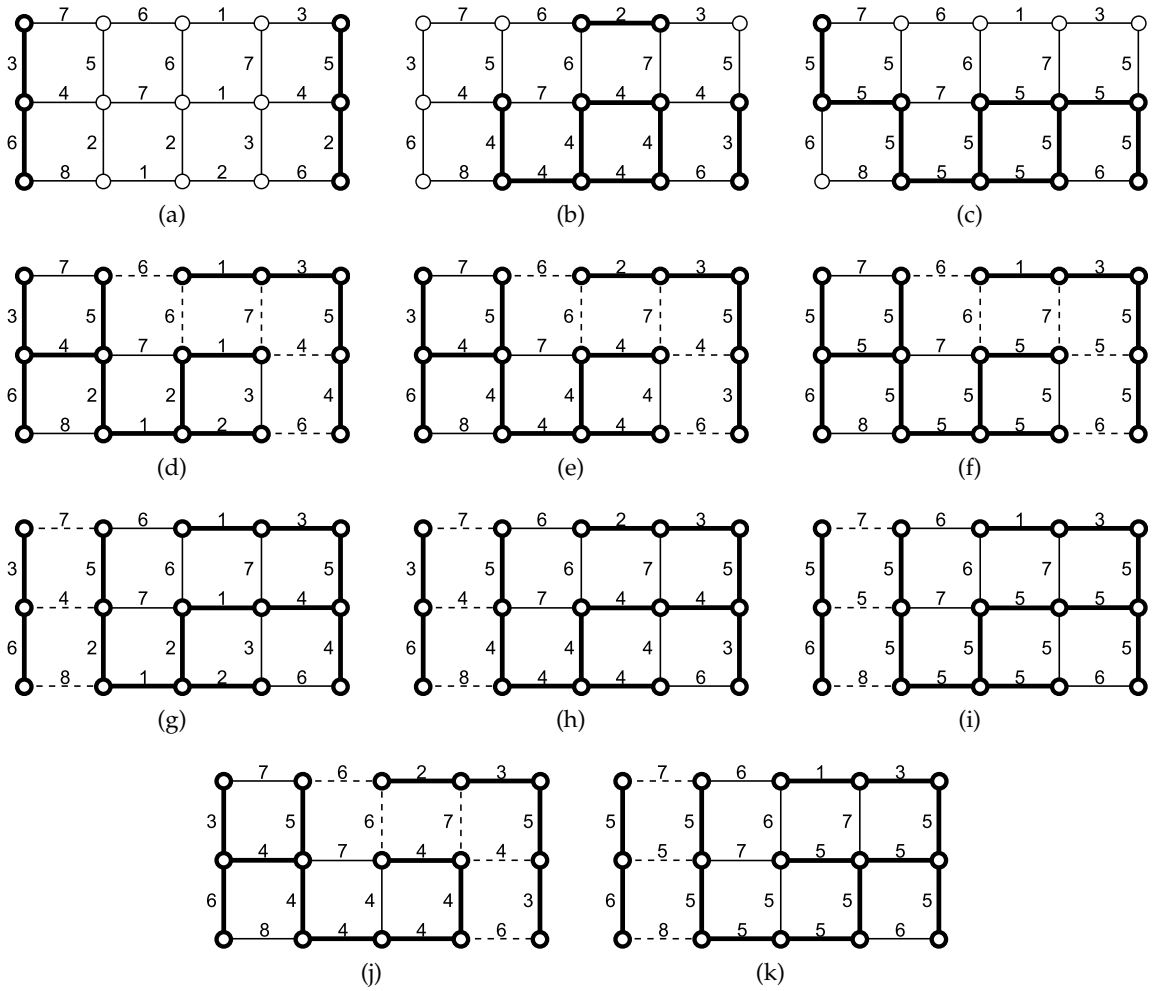


FIGURE 4.9 – Graphe G avec : (a) une application de poids P et, en gras, un sous-graphe M ; (b,c) inondations, respectivement P' et P'', de P (avec les arêtes de poids différents en gras) ; (d) à (f) FCMin F_1 relative à M sur G pour, respectivement, P, P' et P'' ; (g) à (i) FCMin F_2 relative à M sur G pour, respectivement, P, P' et P'' ; (j,k) FCMin relative à M sur G pour, respectivement, P' et P'' mais pas pour P. Les arêtes des coupes induites par les FCMin sont représentées en pointillés.

Remarque 4.18.
Il est à noter que la réciproque du théorème 4.17 n'est, en général, pas vraie (voir figures 4.9j et 4.9k).
Ceci rejoint les remarques 4.9.

Le théorème 4.17 nous permet de trouver le théorème suivant.

Théorème 4.19.
Soit M un sous-graphe de G et $C \subseteq A$ un sous-ensemble d'arêtes.
Si l'ensemble d'arêtes C est une coupe par FCMin relative à M (pour P), alors C est une LPE relative à M (pour P).

Preuve du théorème 4.19.
Supposons que C est une coupe par FCMin relative à M pour P. Considérons P' et P'' tels que définis dans la définition 3.56.
Puisque changer les poids des arêtes de M n'affecte pas les propriétés des FCMin, C est donc une coupe par FCMin relative à M pour P'. On déduit, du théorème 4.17, que C est donc également une coupe par FCMin relative à M pour P'' et, du théorème 4.16, que C est bien une LPE relative à M pour P. □

Une illustration de ce théorème se trouve dans la figure 4.10.

Remarque 4.20.
Il est à noter que la réciproque du théorème 4.19 n'est, en général, pas vraie (voir figures 4.10g à 4.10k).

Comme mentionné dans la section 3.4, une LPE relative calculée sur une inondation restreinte maximale peut très bien passer sur un bassin versant ou un minimum régional de l'application de poids originale (voir figures 4.10g à 4.10k), ce qui, dans la plupart des cas, n'est pas souhaité. Au contraire, une FCMin ne peut donner un tel résultat. Ainsi, dans la figure 3.29 de la section 3.4, seuls les résultats des figures 3.29f et 3.29g peuvent être obtenus si l'on ne considère que les LPE calculées avec une FCMin.

Comme expliqué dans la section 4.3, n'importe quel algorithme de calcul d'un arbre couvrant de poids minimum peut être employé pour déterminer une FCMin et, par conséquence du théorème 4.19, une LPE (relative ou non).
Nous avons vu dans la section 3.3 qu'il existait des algorithmes linéaires pour calculer une LPE relative aux minima régionaux, cependant, dans le cas d'une LPE relative, il faut procéder au calcul préalable de l'inondation maximale contrainte par les marqueurs. Ces deux étapes peuvent donc être avantageusement remplacées par l'unique recherche d'une FCMin sur l'application de poids d'origine afin de prendre sa coupe induite comme résultat de LPE. Certes, il n'y a pas d'algorithme linéaire dans ce cas, mais l'algorithme quasi-linéaire présenté dans [48] est d'une grande rapidité également.

Il est à noter que ce résultat trouve également son intérêt dans le cadre des méthodes de segmentation hiérarchique (voir [145, 149, 156]).

Cette section a ainsi mis en évidence que la recherche d'une LPE, relative ou non, peut toujours se faire par la simple recherche d'une coupe induite par FCMin sur l'application de poids d'origine. Dans le cas d'une LPE relative, la reconstruction morphologique n'est alors plus nécessaire et, de plus, cette méthode évite d'obtenir une segmentation indésirable qui pourrait passer par un minimum régional. Dans ces conditions, il est ainsi préférable de remplacer les algorithmes spécifiques de LPE par un algorithme de calcul de coupe par FCMin.

Preuve du théorème 4.17

Pour prouver le théorème 4.17, nous allons utiliser une caractérisation bien connue des arbres couvrants de poids minimum sur un graphe connexe (propriété 4.21) pour obtenir une caractérisation des FCMin dans un graphe quelconque (propriété 4.22). Nous

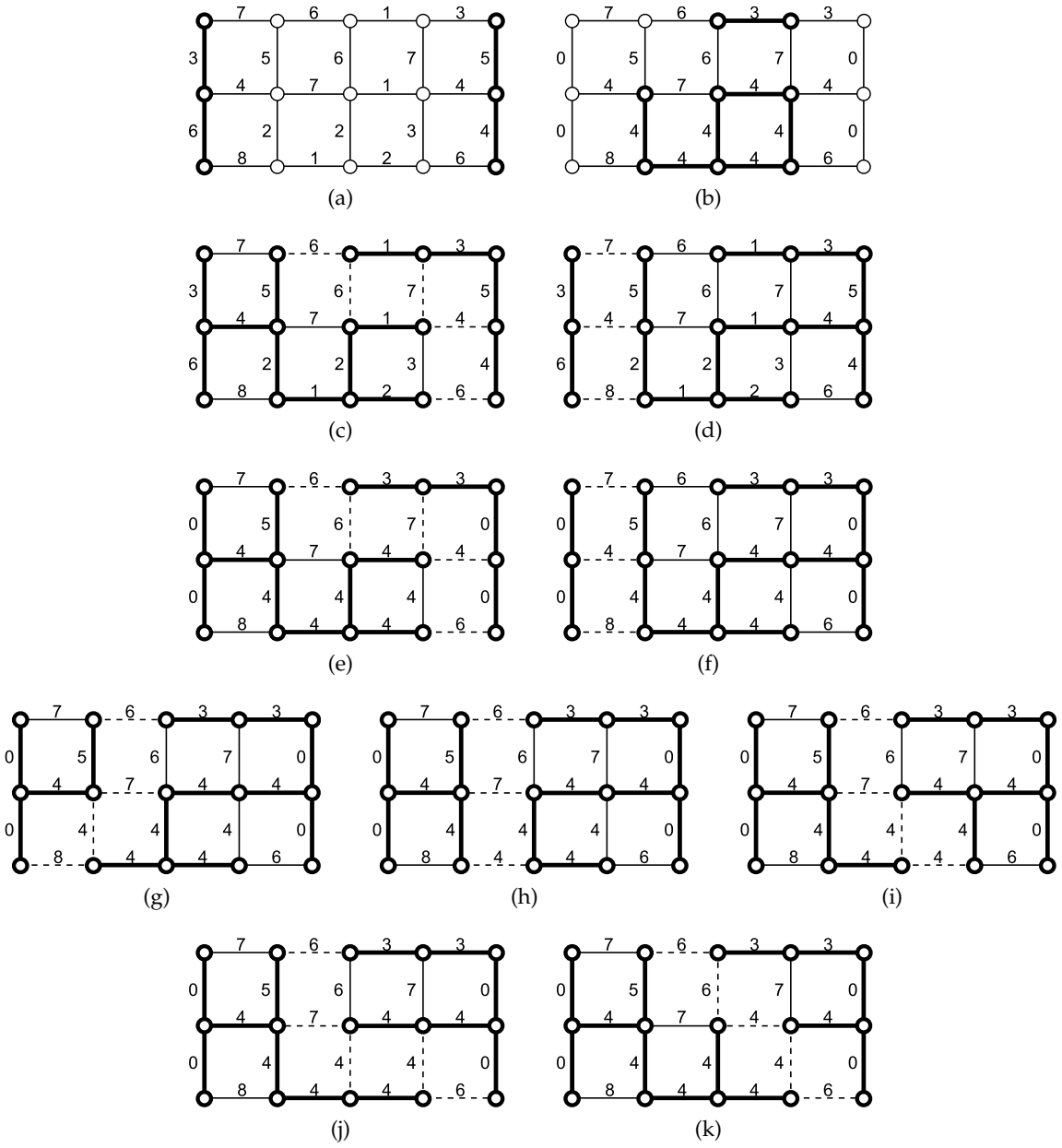


FIGURE 4.10 – Graphe G avec : (a) une application de poids P et, en gras, un sous-graphe M ; (b,c) applications de poids P' et P'', respectivement, telles que décrites dans la définition 3.56 (avec les arêtes de poids différents en gras) ; (d,e) respectivement F_1 et F_2 qui sont deux FCMin relatives à M sur G pour P ; (f,g) respectivement F_1 et F_2 qui sont deux FCMin relatives à M sur G pour P'' également ; (h) à (l) FCMin relatives à M sur G pour P'' mais pas pour P. Les arêtes des coupes induites par les FCMin, qui sont également des LPE relatives à M, sont représentées en pointillés.

montrons enfin que cette caractérisation est préservée au travers des inondations (proposition 4.23).

Considérant que le graphe G est connexe, soit T un arbre couvrant sur G. Pour toute arête $a = \{x, y\} \in (A \setminus A(T))$, on note $\pi_{(T,a)}$ l'unique chaîne élémentaire incluse dans T telle que $\pi_{(T,a)} = \langle x, y, \ldots, x \rangle$.

Propriété 4.21 ([128], chapitre 6.1)**.**
L'arbre couvrant T sur G est un arbre couvrant de poids minimum sur G pour P si et seulement si $\forall a \in (A \setminus A(T))$, $P(a) = H_P(\pi_{(T,a)})$.

Soit F une forêt couvrante G. Pour toute arête $a = \{x, y\} \in (A \setminus A(F))$, on note $\pi_{(F,a)}$ l'unique chaîne élémentaire incluse dans $F \setminus M$ telle que :
- soit $\pi_{(F,a)} = \langle x, y, \ldots, x \rangle$, ou
- soit $\pi_{(F,a)} = \langle m_1, \ldots, x, y, \ldots, m_2 \rangle$ avec $m_1, m_2 \in S(M)$.

Comme on l'a vu dans la section 4.3 une FCMin peut être considérée comme équivalente à un arbre couvrant de poids minimum. Par conséquent, les propriétés des FCMin se déduisent directement de celles d'arbres couvrants de poids minimum. C'est le cas pour la propriété suivante qui découle ainsi de la propriété 4.21.

Propriété 4.22.
Soit M un sous-graphe de G.
La forêt couvrante F sur G est une FCMin relative à M sur G pour P si et seulement si $\forall a \in (A \setminus A(F))$, $P(a) = H_P(\pi_{(F,a)})$.

Cette dernière propriété est préservée au travers des inondations, comme le montre la propriété 4.23 appliquée aux chaînes $\pi_{(F,a)}$ avec $a \in (A \setminus A(F))$.

Propriété 4.23.
Soit E le type d'élément sur lequel se fait la pondération avec l'application de poids P. Soit $\pi = \langle e_0, \ldots, e_{\ell-1} \rangle$ une chaîne dans E et soit $e \in \pi$ tel que $P(e) = H_P(\pi)$.
Si P' est une inondation de P pour G, alors $P'(e) = H_{P'}(\pi)$.

Preuve de la propriété 4.23.
Prouvons ce résultat par l'absurde.
Considérons que P' est une inondation de P telle qu'il existe un élément $e' \in \pi$ tel que $P'(e') > P'(e)$. Il existe alors une valeur $h \in \mathbb{R}$ tel que $P'(e') > h > P'(e)$. Ainsi, la composante connexe X de $G^h_{P'}$ qui contient e de contient pas e'. Puisque P' est une inondation de P, X est donc une composante connexe de G^h_P. Puisque tout élément $\tilde{e} \in \pi$ satisfait $P(\tilde{e}) \leq P(e) \leq P'(e) < h$, tous les éléments de π sont donc dans X, y compris, en particulier, e'. Ce qui contredit l'hypothèse. □

Par conséquent, si F est une FCMin relative à M sur G pour P, alors F est une FCMin relative à M sur G pour P'.

Remarque 4.24.
Il est à noter que l'on déduit directement de la propriété 4.23 qu'une FCCMA sur P est une FCCMA sur l'inondation P' de P, ce qui rejoint une partie du théorème 4.14.

4.8 Récapitulatif et conclusion

Ce chapitre nous a permis de présenter deux définitions de forêts couvrantes optimales, les forêts couvrantes de poids extremum et les forêts couvrantes de chemins de moindre altitude, qui peuvent donc également être considérés comme des paradigmes de segmentation de graphe puisqu'une forêt couvrante induit une coupe unique, offrant

par la même occasion une partition des sommets du graphe (comme déjà vu dans le chapitre 2).

Nous avons également vu dans ce chapitre les relations suivantes, nouvelles pour certaines, entre FCMin, FCMax et FCCMA, où les forêts couvrantes considérées sont relatives à un sous-graphe M sur le graphe G pondéré sur les arêtes par P mais également avec la LPE d'arêtes relative du chapitre 3.

- Toute FCMin est une FCMax, et réciproquement, par l'application d'une fonction strictement décroissante sur P.
- Toute FCMin est une FCCMA et donc toute coupe par FCMin est une coupe par FCCMA.
- Si $M = Min(P)$, FCMin et FCCMA sont équivalentes et donc coupes par FCMin et FCCMA sont équivalentes.
- Coupe par FCCMA et LPE sont équivalentes.
- Toute coupe par FCMin est une LPE.
- Si $M = Min(P)$, coupe par FCMin, coupe par FCCMA et LPE sont équivalentes.

Ainsi, nous pouvons nous rendre compte que coupe par FCMin, coupe par FCMax, coupe par FCCMA et LPE d'arêtes sont toutes liées par certains critères ou bien équivalentes, comme la coupe par FCCMA et la LPE.

L'étude des liens unissant LPE et FCMin a ainsi mis en évidence que, dans le cas d'une recherche de LPE relative, il est préférable que sa recherche se fasse par le biais d'un algorithme de recherche de coupe par FCMin, de complexité quasi-linéaire (voir [48]), plutôt que par l'utilisation d'un algorithme de LPE habituel, certes linéaire (voir la section 3.3, mais nécessitant une transformation préalable de la pondération du graphe (une inondation restreinte maximale, comme présenté dans la section 3.4). De plus, cette méthode a le gros avantage d'empêcher les résultats dont la segmentation passe sur un minimum régional ou sur un bassin versant (tel, par exemple, le résultat présenté en 3.29h et 3.29k de la figure 3.29).

Nous verrons dans le chapitre 5 que des liens avec d'autres critères de segmentation de graphe existent également.

Coupe minimale et flot maximal 5

Après avoir étudié les LPE et les coupes par forêts couvrantes comme moyens de segmentation de graphes dans les chapitres 3 et 4, nous présentons dans ce chapitre les concepts de réseaux de transport et de flot maximal liés à la recherche de coupe minimale qui constitue un autre paradigme 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 ou une denrée 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.

Nous commençons ce chapitre par définir formellement ce qu'est un réseau de transport et un flot dans le cadre des graphes.

Nous abordons 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 présentons la façon de calculer un flot maximal, tout d'abord avec des algorithmes par chemins améliorants, tel celui de Lester Randolph Sr Ford et Delbert Ray Fulkerson, qui furent les premiers à résoudre ce problème en 1956 [85], puis avec un type d'algorithme reposant sur des pré-flots (d'autres algorithmes peuvent également être trouvés dans [3]).

Le problème est ensuite étendu pour la recherche de coupes minimales segmentant un graphe non-orienté en plus de deux régions.

Enfin, nous mettons en évidence le lien existant entre coupe minimale et coupe par FCMax (cette dernière ayant été introduite dans le chapitre 4) et qui constitue la principale nouveauté de ce chapitre.

Les notions traitées dans ce chapitre concernent principalement les graphes orientés. C'est pourquoi nous considérons le graphe orienté connexe $G = (S, A)$, à l'exception de la section 5.7 où nous considérons qu'il est non-orienté. 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 manuscrit, 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.

Sommaire du chapitre

Figures du chapitre

5.1 Réseau de transport et flot

5.1.1 Réseau de transport

Un **réseau de transport** est un graphe orienté à arcs valués par l'application de poids $P : A \to \mathbb{R}^+ \cup +\infty$ 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 5.1.

5.1.2 Flot

On appelle **flot** (compatible avec le réseau de transport $\mathcal{R} = (S, A, P, s, p)$) une application, notée f, de A^* dans $\mathbb{R}$ ($f : A^* \to \mathbb{R}$; 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 1.2.1.2), qui satisfait les trois propriétés suivantes :
- **Contrainte de capacité** :
  si $a \in A$, alors
  $$f(a) \leq P(a) \tag{5.1}$$
  et si $a \in A^* \setminus A$, alors
  $$f(a) \leq 0. \tag{5.2}$$
- **Symétrie** :
  si $a \in A^*$, alors
  $$f(a) = -f(a^{-1}). \tag{5.3}$$
- **Conservation du flot** :
  si $x \in S \setminus \{s, p\}$, alors
  $$\sum_{a=(x,.), a \in A^*} f(a) = 0. \tag{5.4}$$

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 5.1.
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 $x \in S$:

$$f^+(x) = \sum_{a=(.,x), a\in A, f(a)>0} f(a); \tag{5.5}$$

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

$$f^-(x) = \sum_{a=(x,.), a\in A, f(a)>0} f(a). \tag{5.6}$$

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

$$f^+(x) = f^-(x). \tag{5.7}$$

Remarque 5.2.
*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 1.2.1.2) 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 5.1.

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

$$|f| = \sum_{a=(s,.), a\in A^*} f(a) = \sum_{a=(.,p), a\in A^*} f(a) \tag{5.8}$$

Des exemples de flots compatibles avec un réseau de transport se trouvent dans la figure 5.1.

Remarque 5.3.
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 5.4 (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,
– des voitures sur un réseau routier,
– 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.

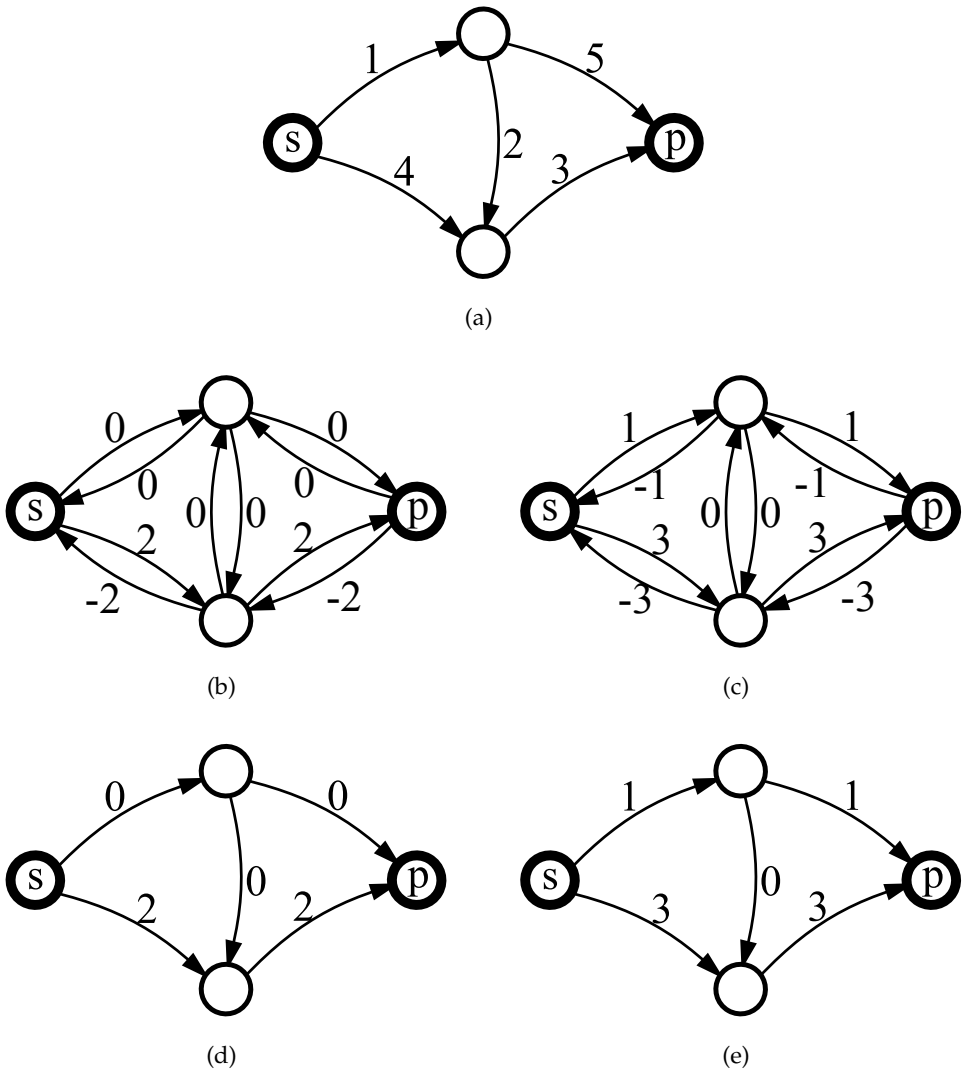


FIGURE 5.1 – (a) Réseau de transport $\mathcal{R}$ avec la source s et le puits p représentés en gras. (b) Flot f_1 compatible avec $\mathcal{R}$. (c) Flot maximal f_2 compatible avec $\mathcal{R}$. (d) Flot net positif de f_1. (e) Flot net positif de f_2.

5.2 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 5.5 (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 $y \in \mathcal{T}\}$ 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.*

Cette définition n'est pas sans rapport avec la coupe dans un graphe non-orienté présentée dans la définition 2.1 puisqu'on peut y retrouver, dans sa seconde version, la notion d'extension maximale relative à $\{\{s, p\}, \emptyset\}$.

Malgré cela, 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 5.2. 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 5.6.

Propriété 5.6.
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 5.2.

Remarque 5.7.
Outre la propriété 5.6 qui lie coupe relative à un sous-graphe (définition 2.1) et coupe d'un réseau (définition 5.5), 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. Nous verrons plus loin, dans la section 5.6, que cela peut être étendu à plus de deux régions, permettant ainsi d'utiliser la recherche de coupe minimale comme méthode de segmentation de graphe (voir chapitre 2).

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

$$P(C) = \sum_{a \in C} P(a). \tag{5.9}$$

On appelle **coupe minimale** d'un réseau de transport une coupe de poids minimal.

Le théorème suivant, prouvé dans [67] et présenté dans [70] (chapitre 6.2), montre le lien qui unit la recherche d'une coupe minimale avec la recherche d'un flot maximal.

Théorème 5.8.
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 5.10.

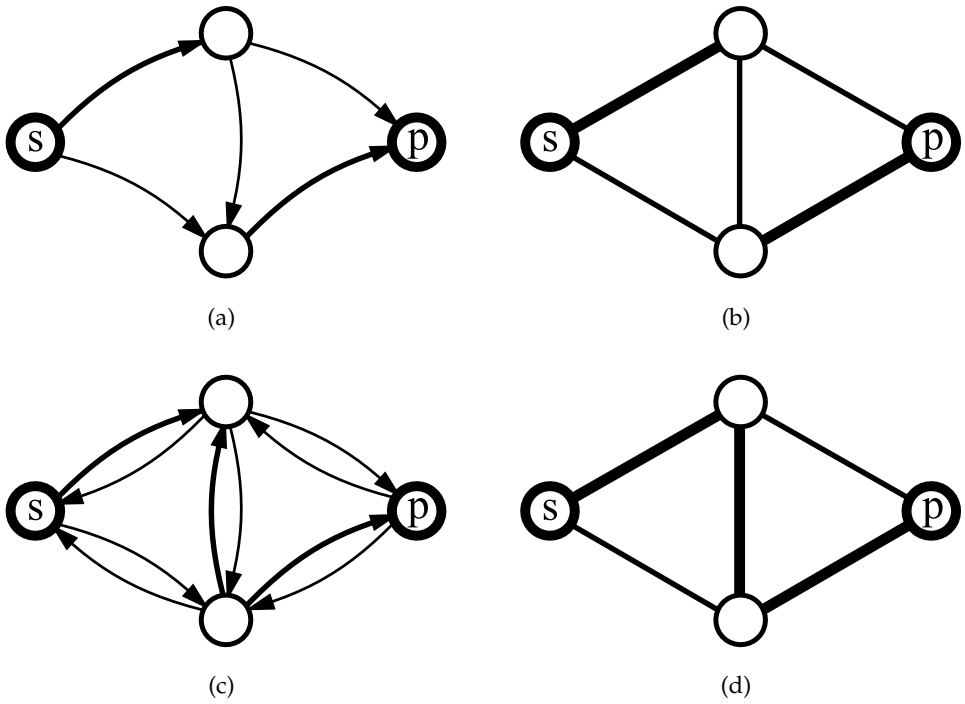


FIGURE 5.2 – (a) Réseau $\mathcal{R}_1 = (S, A_1, P_1, s, p)$ avec, en gras, une coupe C_1. (b) Graphe non-orienté associé à $\mathcal{R}_1$ avec, en gras, les arêtes associées à C_1 qui ne sont pas une coupe relative à $(\{s, p\}, \emptyset)$. (c) Réseau $\mathcal{R}_2 = (S, A_2, P_2, s, p)$ avec, en 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 5.9.
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.

5.3 Recherche de flot maximal

Comme nous l'avons vu dans la section 5.2, 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 [91]. Les livres [52, 185] peuvent également être recommandés pour une étude plus détaillée de ces méthodes. Celles-ci se décomposent en deux grandes catégories qui sont par chemins améliorants et par poussée de flot, présentées respectivement dans les sous-sections 5.3.1 et 5.3.2.

5.3.1 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 [85, 86], 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.

5.3.1.1 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 :

$$r(a) = \begin{cases} P(a) - f(a) & \text{si } a \in A; \\ -f(a) & \text{sinon.} \end{cases} \tag{5.10}$$

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 :

$$A_f = \{a \in A^* \mid r(a) > 0\}. \tag{5.11}$$

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 5.5e).

Des exemples de réseaux résiduels se trouvent dans la figure 5.3.

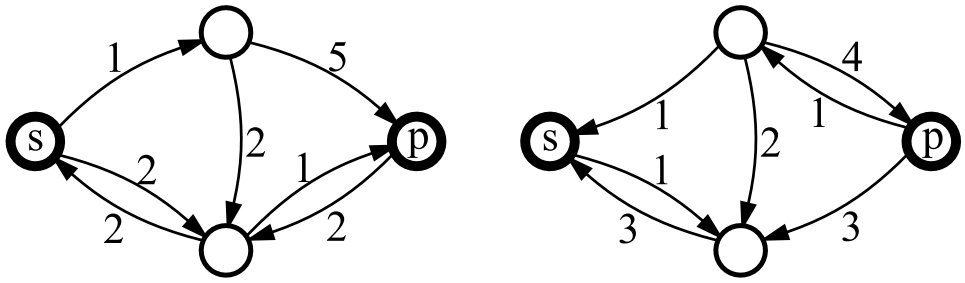


FIGURE 5.3 – (a) Graphe résiduel de f_1 sur $\mathcal{R}$ (figure 5.1). (b) Graphe résiduel de f_2 sur $\mathcal{R}$ (figure 5.1). La source et le puits sont représentés en gras.

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 5.10.

Théorème 5.10.
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 5.11 (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 5.12.
Un flot maximal est un flot complet.

Il est à noter que la réciproque du théorème 5.12 n'est, en général, pas vraie.
Une illustration de cela se trouve dans la figure 5.4.

5.3.1.2 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 $\mathcal{R}$.

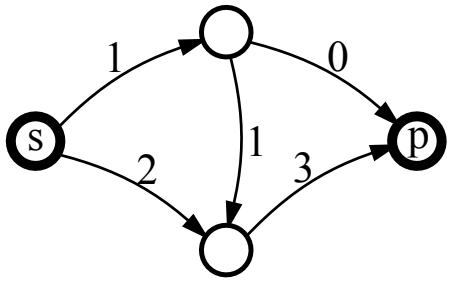


FIGURE 5.4 – Flot complet, mais non maximal, compatible avec le réseau de transport de la figure 5.1a.

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 5.12 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 $r(\pi)$, 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 :

$$r(\pi) = \min_{a \in \pi} \{r(a)\}. \tag{5.12}$$

5.3.1.3 Algorithme de Ford et Fulkerson

L'algorithme de Ford et Fulkerson [85, 86] 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 $r(\pi)$. Retourner à l'étape 1.

Un exemple de déroulement de cet algorithme se trouve dans la figure 5.5.

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 [86, 185].

Le plus souvent, cet algorithme est utilisé avec des capacités entières. On obtient alors une complexité en $O(|A||\tilde{f}|)$ 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.

5.3.1.4 Variantes et améliorations

Comme dit précédemment, la stratégie de sélection du chemin améliorant est déterminant 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.

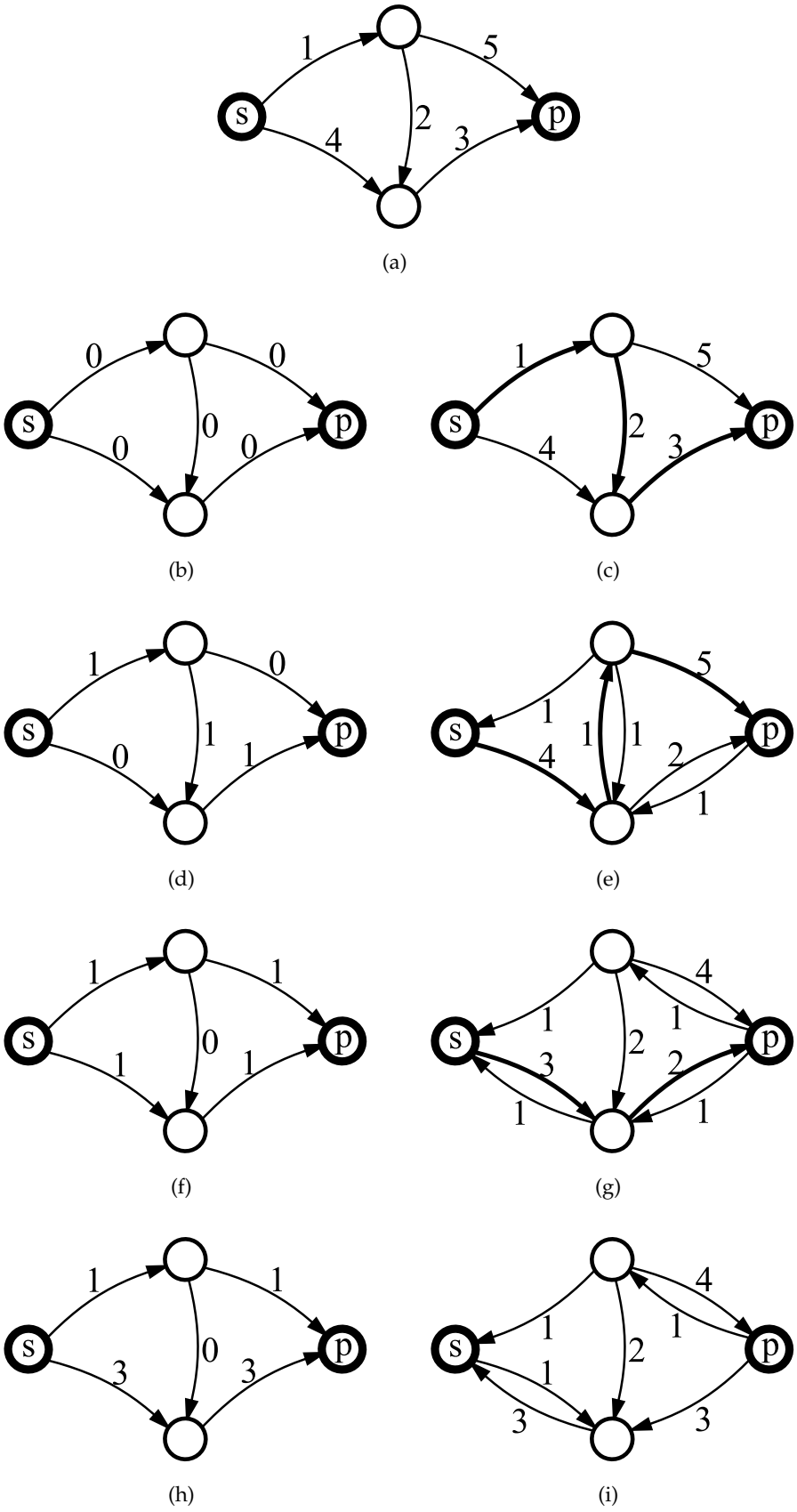


FIGURE 5.5 – (a) Réseau de transport $\mathcal{R}$. (b,d,f,h) Flots nets positifs de, respectivement f_0, f_1, f_2, f_3, compatibles avec $\mathcal{R}$. (c,e,g,i) Graphes résiduels de, respectivement f_0, f_1, f_2, f_3, sur $\mathcal{R}$. Dans (c,e,g), un chemin améliorant est représenté en gras.

En 1970, E. A. Dinic proposa un algorithme [72] basé sur une recherche de chemins améliorants par un parcours en largeur (voir [53], chapitre 23). 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 [78]. 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 (voir la sous-section 5.3.2), 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 [32, 33] (voir chapitre 6). 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 $O(|A||S|^2|\tilde{f}|)$, 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 (illustrant ainsi les propos tenus en fin de section 1.3.1). C'est, depuis lors, cet algorithme qui fait office de référence en vision par ordinateur.

5.3.2 Recherche de flot maximal par poussée de flot

La seconde grande catégorie d'algorithmes de recherche de flot maximal est dite par poussée de flot (*push-relabel* en anglais). Il n'est plus ici question de partir d'un flot et de l'améliorer itérativement, au travers d'un chemin reliant source et puits, jusqu'à obtenir le flot maximal mais d'envoyer le maximum de flot qu'un sommet a reçu vers ses voisins en direction du puits. Il n'est ainsi pas rare d'avoir trop envoyé de flot dans certains arcs si celui-ci ne peut pas être propagé plus loin dans le réseau. Le flot net y transitant doit alors être abaissé ultérieurement.

Il est à noter que nous n'avons dès lors plus affaire à un flot, tel que défini dans la section 5.1.2, mais à ce qui s'appelle un **pré-flot**, comme nous allons le voir ci-après.

5.3.2.1 Pré-flot

On appelle **pré-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 encore que $A^* = A \cup A^{-1}$ - voir section 1.2.1.2), qui satisfait les trois propriétés suivantes :

– **Contrainte de capacité** :
si $a \in A$, alors

$$f(a) \leq P(a) \tag{5.13}$$

et si $a \in A^* \setminus A$, alors

$$f(a) \leq 0. \tag{5.14}$$

– **Symétrie** :
si $a \in A^*$, alors
$$f(a) = -f(a^{-1}). \tag{5.15}$$

– **Accumulation du flot** :
si $x \in S \setminus \{s\}$, alors
$$\sum_{a=(x,.),a\in A^*} f(a) \geq 0. \tag{5.16}$$

On observe donc que la seule différence entre un flot et un pré-flot est que la propriété de conservation du flot (équation 5.4) est remplacée par une version "relâchée" qui est celle d'accumulation de flot (équation 5.16). De plus, pour tout sommet $x \in S \setminus \{s\}$, on appelle **excédent de flot** (du sommet x pour le pré-flot f), noté $e_f(x)$ la quantité de flot accumulée en ce sommet :
$$e_f(x) = \sum_{a=(x,.),a\in A^*} f(a). \tag{5.17}$$

La source est le seul sommet à émettre du flot sans en recevoir. On a donc, d'après la propriété d'accumulation du flot :
$$\forall x \in S \setminus \{s\}, e_f(x) \geq 0. \tag{5.18}$$

On dit qu'un sommet $x \in S \setminus \{s\}$ **déborde** si $e_f(x) > 0$.

Remarque 5.13 (Pré-flot et représentation)**.**
Comme dans la remarque 5.2 concernant la représentation d'un flot, la représentation d'un pré-flot, de par sa définition, doit également se faire sur la fermeture symétrique (voir la définition en section 1.2.1.2) du graphe du réseau de transport. Toutefois, pour des raisons de clarté de représentation, on peut se contenter de représenter les arcs du graphe d'origine uniquement.
Voir des exemples sur la figure 5.6.

Des exemples de pré-flots compatibles avec un réseau de transport se trouvent dans la figure 5.6.

5.3.2.2 Algorithme de *push-relabel*

Soit h une application, appelée **hauteur** (au puits), de S dans $\mathbb{N}$ approximant la hauteur, en terme de longueur de chemin, de chaque sommet par rapport au puits. Cette hauteur est amenée à évoluer durant l'exécution de l'algorithme. Une initialisation possible de cette hauteur est, pour tout sommet $x \in S \setminus \{s\}$, de mettre $h(x) = 0$ et, pour la source, de mettre $h(s) = |S|$.

Un algorithme de *push-relabel* consiste à mettre à jour un pré-flot et une application de hauteur au travers d'itérations de deux opérations simples :

1. **Poussée** (*push* en anglais) : le flot en trop du sommet qui déborde le plus haut par rapport au puits est poussé vers son sommet voisin le moins haut. Si cela n'est pas possible, alors on a un flot maximal.
2. **Ré-étiquetage** (*relabel* en anglais) : chaque sommet qui déborde et n'ayant aucun sommet voisin avec une hauteur strictement inférieure à la sienne voit sa hauteur remplacée par $h + 1$, où h est la hauteur la plus basse de ses voisins.

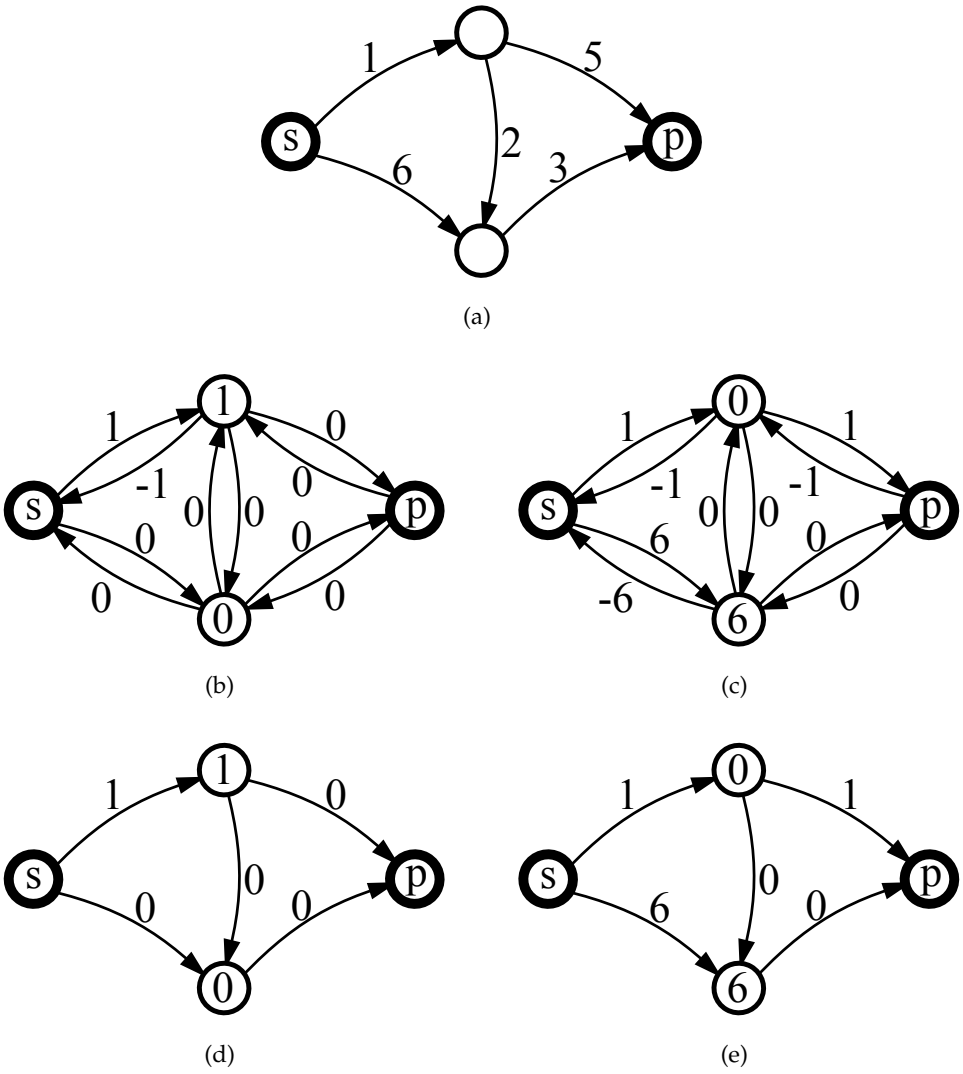


FIGURE 5.6 – (a) Réseau de transport $\mathcal{R} = (S, A, P, s, p)$. (b) Pré-flot f_1 compatible avec $\mathcal{R}$. (c) Pré-flot f_2 compatible avec $\mathcal{R}$. (d) Représentation de f_1 sur (S, A). (e) Représentation de f_2 sur (S, A). Les applications d'excédent de flot e_{f_1} et e_{f_2} sont représentées sur les sommets des pré-flots correspondants autres que la source s et le puits p.

Ainsi, contrairement à la méthode par chemin améliorant, chaque étape de poussée est localisée puisque l'on ne s'intéresse qu'à un seul sommet.

Il est à noter que, avec l'augmentation de la hauteur des sommets débordants qui n'ont pas de voisin avec une hauteur strictement inférieure, on est assuré que le flot sera toujours propagé vers des sommets étiquetés comme étant plus bas. De plus, l'excédent de flot qui peut demeurer une fois que le puits a été atteint est automatiquement renvoyé vers la source, donnant bien pour résultat final un flot, qui est maximal, et non pas un pré-flot.

5.3.2.3 Variantes et améliorations

Ce fut Alexander V. Karzanov qui introduit la concept de pré-flot en 1974 [116]. Il proposa un algorithme de recherche de flot maximal dans les graphes denses (i.e. les graphes dont le nombres d'arcs est proche du nombre maximal possible, autrement dit, un graphe "presque" complet) ayant une complexité en $O(|S|^2)$. Bien que l'étape de poussée ne soit qu'implicite dans son algorithme, celui-ci peut être considéré comme le premier de type *push-relabel*.

En 1985, Andrew V. Goldberg proposa un algorithme de *push-relabel* de complexité en $O(|S|^3)$. Robert E. Tarjan l'aida à l'améliorer à partir de 1986, notamment grâce à l'emploi de structures de données par arbre dynamique (voir [192, 193]) qui leur permirent d'obtenir une complexité en $O\big(|S||A|\log(\frac{|S|^2}{|A|})\big)$ [92, 93, 94].

En 1987, Ravindra K. Ahuja et James B. Orlin trouvèrent un algorithme de complexité en $O\big(|S||A| + |S|^2\log(p_{\max})\big)$, où $p_{\max}$ est la capacité la plus élevée de tous les arcs de A ($p_{\max} = \max_{a\in A} P(a)$), en réintroduisant l'approche de E. A. Dinic dans le cadre du *push-relabel* [4]. Avec l'aide de Robert E. Tarjan, ils l'améliorèrent pour obtenir un algorithme de complexité en $O\big(|S||A| + |S|^2\sqrt{\log(p_{\max})}\big)$ qui leur permit également d'utiliser les structures de données par arbre dynamique, offrant alors une complexité en $O\big(|S||A|\log\big(\frac{|S|}{|A|}\sqrt{\log(p_{\max})}+2\big)\big)$ [5, 6].

Les meilleures complexités d'algorithmes de *push-relabel* connues à ce jour sont donc celles en $O\big(|S||A|\log(\frac{|S|^2}{|A|})\big)$ dans [92, 93, 94] et celles en $O\big(|S||A|\log\big(\frac{|S|}{|A|}\sqrt{\log(p_{\max})}+2\big)\big)$ dans [5, 6]. Ce sont également les meilleures complexités obtenues pour tout type d'algorithme de recherche de flot maximal.

5.3.3 Méthodes alternatives sur réseaux dynamiques

Nous entendons par réseaux dynamiques des réseaux qui évoluent en termes de capacités d'arcs mais qui peuvent aussi, selon les cas, subir des modifications de topologie. Les deux méthodes présentées ci-après sont adaptées à ce cas, leur permettant d'obtenir des résultats plus rapides que de simples exécutions des algorithmes précédents à chaque itération.

5.3.3.1 *Dynamic cuts*

En 2005, Pushmeet Kohli et Philip H.S. Torr développèrent les *dynamic cuts* [118, 119] qui permettent de calculer, à partir d'un flot maximal dans un réseau donné $\mathcal{R}$, un flot maximal d'un réseau $\mathcal{R}'$ différent de $\mathcal{R}$ par les capacités de certains arcs uniquement. Le calcul du nouveau flot se fait en temps linéaire par rapport au nombre d'arcs dont la capacité a été modifiée.

Cette méthode fut créée pour trouver dynamiquement les coupes minimales d'un graphe mis à jour en temps réel pour faire de la segmentation dans des séquences vidéo. Toutes les images de la vidéo sont autant d'images de tailles identiques qui peuvent donc être associées à un graphe orienté à arcs valués (voir 1.5.3), mais au lieu de faire une segmentation séparée image par image, il est alors possible de se servir du résultat de l'une d'elle pour calculer plus rapidement celui de la suivante. Ceci permet donc de faire une segmentation en temps réel d'un objet par rapport à son arrière-plan sur une vidéo par exemple.

5.3.3.2 *Active cuts*

En 2005 également, Olivier Juan et Yuri Boykov développèrent les *active cuts* [112, 113] qui permettent de calculer, à partir d'une coupe (typiquement une coupe minimale) dans un réseau donné $\mathcal{R}$, une coupe minimale d'un réseau $\mathcal{R}'$ qui peut également différer de $\mathcal{R}$ par les capacités de certains arcs mais aussi par sa topologie. Par rapport aux *dynamic cuts*, le changement de topologie est ici possible car on ne se base plus sur un flot pour calculer le suivant, mais tout simplement sur la partition des sommets du réseau.

De façon simplifiée, chaque étape d'*active cuts* consiste en l'exécution d'un algorithme de type *push-relabel* autour de la coupe d'initialisation (ce principe s'inspire du concept de calculs limités aux *narrow band* pour les ensembles de niveaux - voir section 6.3.3). La complexité de cet algorithme est relative à celle de l'algorithme de *push-relabel* utilisé. Ainsi, dans le pire cas, on obtiendra un temps d'exécution plus long. Néanmoins, si l'hypothèse comme quoi des coupes minimales successives sont proches l'une de l'autre est vérifiée, le gain de temps est considérable.

Ainsi, en plus des applications possibles pour les *dynamic cuts*, les *active cuts* permettent, par exemple, de rechercher des coupes minimales à travers une méthode multi-échelles.

Il est à noter qu'un peu avant, dans la même année, une méthode similaire (basée également sur le principe des *narrow bands* liées aux ensembles de niveaux - voir section 6.3.3) fut proposée dans [140] afin de réduire le temps et l'espace requis pour le calcul de coupes minimales. Malheureusement, dans ce cas, la méthode proposée n'était qu'une heuristique et seule une approximation de la coupe minimale globale pouvait être obtenue alors que les *active cuts* garantissent une coupe minimale.

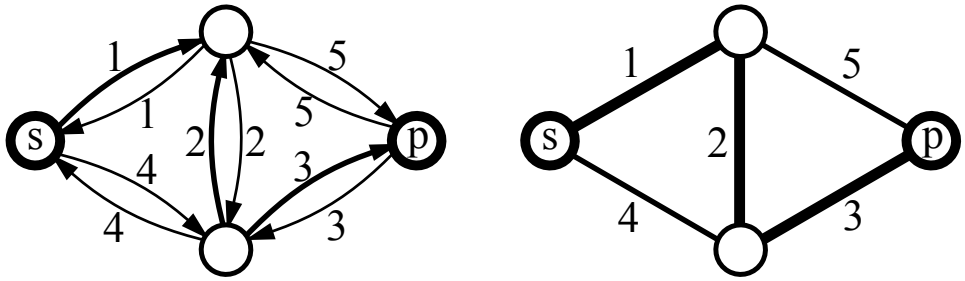
5.4 Coupe minimale dans un graphe non-orienté

Comme expliqué dans la section 1.2.3, 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 5.8 peut ainsi facilement être étendu au cas des graphes non-orientés, comme montré dans [86]. En effet, en se référant à la propriété 5.6, 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 5.7.

Remarque 5.14.
Il est à noter que dans le cas d'une recherche de coupe minimale dans un graphe non-orienté, la



(a) (b)

FIGURE 5.7 – (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 $(\{s, p\}, \emptyset)$.

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.

5.5 Coupe minimale relative

Dans le but de nous rapprocher de la définition de coupe présentée dans le chapitre 2 (définition 2.1), nous montrons qu'il est possible de considérer un sous-graphe M à deux composantes connexes comme marqueur pour le calcul d'une coupe minimale dans un graphe G orienté ou non.

On appelle **coupe minimale relative** à M (sur G) un ensemble d'arcs ou d'arêtes $C \subseteq A$ qui constitue une coupe minimale du réseau de transport $\mathcal{R} = (S', A', P', s, p)$ construit de l'une des deux façons suivantes :

1. en ajoutant une source et un puits reliés chacun aux sommets de composantes connexes de M différentes par des arcs ou des arêtes de poids infini :
   - ajouter aux sommets de S la source s et le puits p qui représenteront chacun une composante connexe de M, ainsi on a $S' = S \cup \{s, p\}$;
   - pour chaque sommet de M, ajouter un arc ou une arête pour le relier au sommet représentant la composante connexe à laquelle il appartient (dans le cas d'un graphe orienté, les arcs ajoutés vont, respectivement, de la source vers le sommet et du sommet vers le puits), ainsi on a $A' = A \setminus \left\{ \cup_{i=1}^{|S(M)|} a_i \right\}$;
   - l'application de poids P' est l'application de poids P à laquelle on ajoute des poids infinis pour chacun des arcs ou arêtes rajoutés.
2. en fusionnant chaque composante connexe de M de sorte à en faire, respectivement, la source ou le puits du nouveau graphe :
   - remplacer les sommets de chaque composante connexe de M par un seul qui sera, respectivement, la source s ou le puits p, ainsi on a $S' = \{S \setminus S(M)\} \cup \{s, p\}$;
   - supprimer les arcs ou les arêtes induites par chaque composante connexe de M et rediriger les arcs ou arêtes adjacent(e)s à chaque composante de M vers le nouveau sommet qui la représente, ainsi on a $A' = A \setminus \left\{ \cup_{i=1}^{|\mathcal{C}(M)|} A(M_i) \right\}$;
   - l'application de poids P' est l'application de poids P à laquelle on a enlevé les poids des arcs ou arêtes retirées.

La première méthode proposée a l'avantage de ne pas modifier le graphe d'origine et de n'y faire que des ajouts, mais on se retrouve alors avec des arcs ou arêtes supplémentaires de poids infinis qui seront susceptibles de ralentir le calcul de la coupe

minimale relative. Au contraire, la seconde méthode supprime certains arcs ou arêtes qui n'interviendront pas dans le calcul de la coupe minimale relative.

Il est à noter que, dans le cas d'un graphe orienté, on sait quelle composante connexe de M représente, respectivement, la source et le puits. Ceci n'est pas nécessaire dans le cas d'un graphe non-orienté comme l'explique la remarque 5.14.

Des illustrations de coupes minimales relatives et des graphes modifiés correspondants se trouvent dans les figures 5.8 et 5.9 pour, respectivement, le cas orienté et non-orienté.

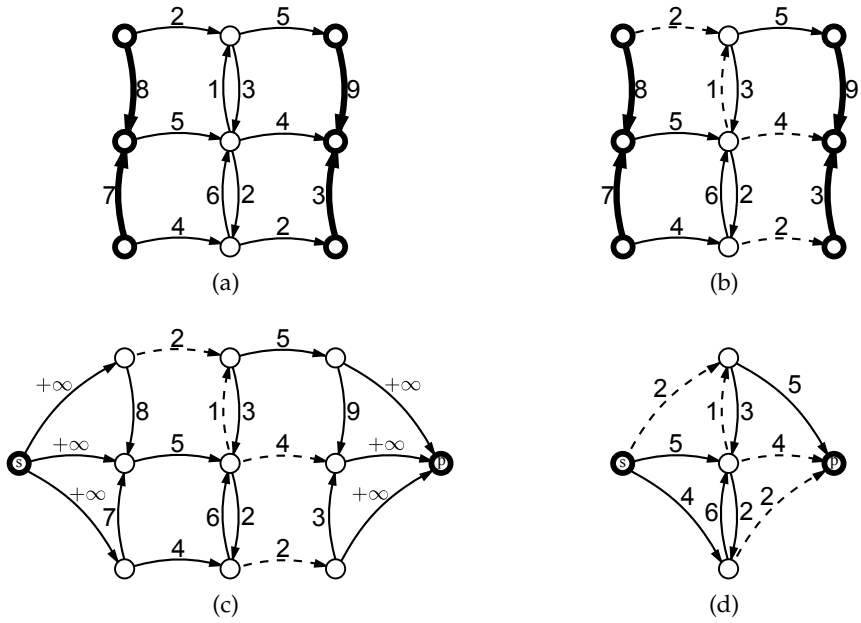


FIGURE 5.8 – (a) Graphe orienté G avec, en gras, un sous-graphe M à deux composantes connexes. (b) Graphe G avec, en gras, le sous-graphe M et, en pointillés, la coupe minimale relative à M. (c) Graphe G_1 obtenu à partir de G par la méthode 1 avec, en gras, les sommets source/puits et, en pointillés, une coupe minimale. (d) Graphe G_2 obtenu à partir de G par la méthode 2 avec, en gras, les sommets source/puits et, en pointillés, une coupe minimale.

Dans la suite de ce mémoire, nous considérerons que les coupes minimales relatives à un sous-graphe seront obtenues par l'une ou l'autre des méthodes décrites ci-avant mais pourront très bien avoir une représentation similaire à celles de la figure 5.8b ou 5.9b uniquement.

Il est à noter que, dans le cas d'un graphe non-orienté, une coupe minimale relative est donc bien une coupe telle que décrite dans la définition 2.1 pour un sous-graphe M ayant exactement deux composantes connexes.

Remarque 5.15.
La section 5.6 étend la notion de coupe minimale à plus de deux régions en se référant non plus à une source et un puits, mais à un ensemble de plus de deux sommets (soit des couples $(s_1, p_1), (s_2, p_2), \ldots, (s_k, p_k)$ comme dans la sous-section 5.6.2, soit des terminaux $\{t_1, \ldots, t_k\}$ comme dans la sous-section 5.6.3).
Il est à noter que les composantes connexes d'un sous-graphe M peuvent, dans ces cas également,

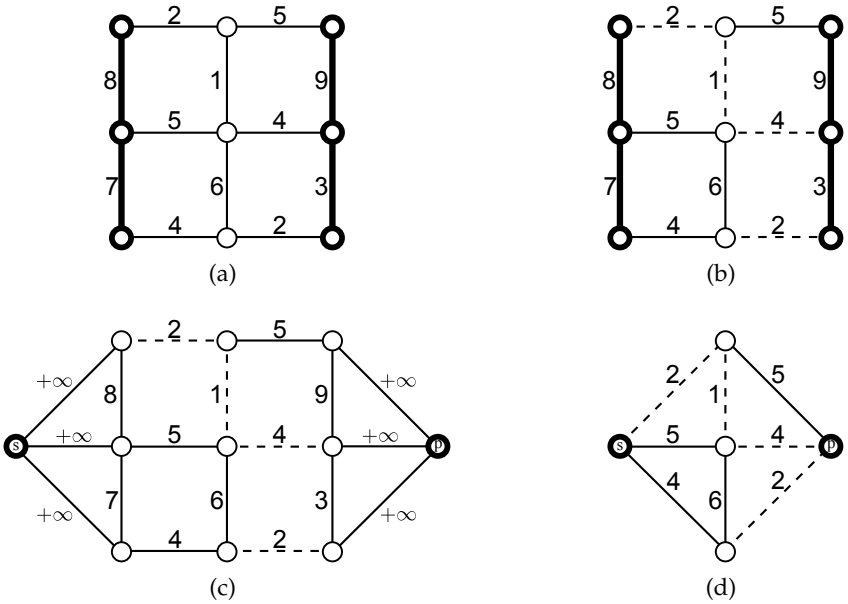


FIGURE 5.9 – (a) Graphe non-orienté G avec, en gras, un sous-graphe M à deux composantes connexes. (b) Graphe G avec, en gras, le sous-graphe M et, en pointillés, la coupe minimale relative à M. (c) Graphe G_1 obtenu à partir de G par la méthode 1 avec, en gras, les sommets source/puits et, en pointillés, une coupe minimale. (d) Graphe G_2 obtenu à partir de G par la méthode 2 avec, en gras, les sommets source/puits et, en pointillés, une coupe minimale.

remplacer aisément chacun de ces sommets particuliers au travers de l'une ou l'autre des constructions proposées ci-avant.

5.6 Recherche de coupe minimale avec plus de deux régions

Comme déjà mentionné dans la remarque 5.7, la recherche de coupe minimale peut être étendue à plus que deux régions, offrant ainsi un outil de segmentation de graphe aux applications plus larges.

Les notions qui suivent s'appliquent dans des graphes non-orientés valués sur les arêtes, devenant ainsi une extension des coupes minimales pour deux régions décrites dans la section 5.4. Nous considérerons ainsi, dans la suite de cette section, le graphe non-orienté $G = (S, A)$ avec l'application de poids P sur les arêtes.

Tout d'abord, bien que légèrement éloigné du problème de coupe minimale, nous introduirons rapidement le problème de k-coupe minimale qui a la spécificité de produire une segmentation en k régions sans avoir de sommets d'initialisation. Nous aborderons ensuite le problème de *multipair cut* minimale qui est une première extension du problème de coupe minimale consistant à minimiser la séparation entre plusieurs couples source-puits au lieu d'un seul. Enfin, nous présenterons le problème de *multiway cut* minimale, qui généralise le problème de coupe minimale à un nombre quelconque de sommets appelés terminaux. Nous décrirons également les méthodes les plus couramment utilisées pour sa résolution.

5.6.1 k-coupe minimale

On appelle **k-coupe**, où k est un entier strictement positif, un ensemble d'arêtes $C \subseteq A$ tel que le graphe dual à A (i.e. $\mathcal{D}(A)$) possède au moins k composantes connexes.

La recherche d'une k-coupe de poids minimal a été montré comme NP-complet [96], cependant, à la différence des problèmes qui vont suivre (qui possèdent des contraintes liés à certains sommets particuliers), celui-ci possède des algorithmes de résolution en temps polynomial (pour une valeur k fixée - voir [95, 183]) avec, pour meilleure complexité $O(|S|^{k^2/2-3k/2+4}T(|S|,|A|))$ où $T(|S|,|A|)$ représente la complexité d'un algorithme de recherche de coupe minimale (voir [96]). Il est à noter que, dans le cas particulier d'un graphe planaire, il existe des algorithmes de meilleure complexité pour la recherche d'une 2-coupe minimale (voir [151]) et celle d'une 3-coupe minimale (voir [97]). Toutefois, il n'y a pas de généralisation pour tout k et, si $k \geq 6$ le meilleur moyen de trouver une k-coupe minimale dans un graphe planaire est d'utiliser l'algorithme de recherche de *multiway cut* minimale de [66] (voir section 5.6.3) pour tous les sous-ensembles de k terminaux (voir [66]).

5.6.2 *Multipair cut* minimale

Soient k paires de sommets $(s_1, p_1), (s_2, p_2), \ldots, (s_k, p_k)$ de S.
On appelle ***multipair cut***, ou ***multicut***, un ensemble d'arêtes $C \subseteq A$ tel que, pour tout $i \in [1;k]$, il n'existe aucune chaîne reliant la source s_i au puits p_i dans le graphe $G' = (S, A \setminus C)$ et aucun sous-ensemble strict $C' \subset C$ ne vérifie cette propriété.

La recherche d'une *multipair cut* de poids minimal fut soulevée par T. C. Hu en 1969 dans [105] (p.150).
Si on a $k = 1$, nous retrouvons alors le problème de la coupe minimale dans un graphe non-orienté.
Si on a $k = 2$, le problème reste soluble en temps polynomial en appliquant deux fois l'algorithme de recherche de coupe minimale [211].
Si on a $k \geq 3$, cette recherche s'avère être NP-difficile [66], en conséquence de quoi, dans la plupart des cas, ne peuvent être trouvées en pratique que des approximations du résultat minimal. Le meilleur algorithme d'approximation à notre connaissance fut proposé en 1996 dans [88].

Remarque 5.16.
Ce problème est à ne pas confondre avec le cas où, pour tout $i, j \in [1;k]$, la source s_i doit être séparée du puits p_j. Dans ces conditions, on peut très facilement obtenir une solution optimale en traitant l'ensemble des sources s_i et, respectivement, l'ensemble des puits p_j comme les sommets de deux composantes connexes dont on voudrait calculer une coupe minimale relative (voir section 5.5).

5.6.3 *Multiway cut* minimale

Soient k sommets $t_1, \ldots, t_k$ de S qu'on appelle **terminaux**.
On appelle ***multiway cut***, ou ***multiterminal cut***, un ensemble d'arêtes $C \subseteq A$ tel que, pour tout $i, j \in [1;k]$ avec $i \neq j$, il n'existe aucun chaîne reliant les terminaux t_i et t_j dans le graphe $G' = (S, A \setminus C)$ et aucun sous-ensemble strict $C' \subset C$ ne vérifie cette propriété.

Remarque 5.17.
Il est à noter que cette définition correspond exactement à celle de coupe décrite dans le chapitre 2 (définition 2.1) dans le cas où le sous-graphe M servant de marqueur n'est composé que de sommets isolés. Si cette dernière condition n'est pas remplie, il est facile de s'y rapporter au travers d'une modification du graphe telle que présentée dans la section 5.5 (voir remarque 5.15).

La recherche d'une *multiway cut* de poids minimal fut soulevée dans [86]. En 1983, ce problème fut repris dans un abstract étendu non publié [64] qui ne fut développé pour être publié qu'une dizaine d'années plus tard dans [65, 66] en reprenant les notations introduites dans [63].

Si on a $k = 2$, nous retrouvons alors le problème de la coupe minimale dans un graphe non-orienté.

Si on a $k \geq 3$, cette recherche s'avère être NP-difficile ([66], théorèmes 2 et 3) et même MAX SNP-difficile ([66], théorème 5), en conséquence de quoi ne peuvent être trouvées en pratique que des approximations du résultat minimal avec, de plus, un choix de ratio d'approximation limité. Une heuristique très simple, appelée heuristique d'isolation, donnant une $\left(2 - \frac{2}{k}\right)$-approximation en $O\left(k|S||A| \log \frac{|S|^2}{|A|}\right)$ (présentée ci-après) se trouve dans [66] (théorème 4). En se basant sur celle-ci, un algorithme donnant une $\left(1,5 - \frac{1}{k}\right)$-approximation fut proposée en 2000 dans [62].

Il est à noter que, dans le cas particulier d'un graphe planaire (i.e. un graphe qui peut se représenter dans un plan sans qu'aucune paire d'arcs/arêtes ne se croise), bien que la recherche d'une *multiway cut* minimale soit toujours NP-difficile ([66], théorème 2), des algorithmes polynomiaux permettent d'obtenir une solution en $O(|S|^3 \log |S|)$ pour $k = 3$ et en $O\left((4k)^k |S|^{2k-1} \log |S|\right)$ pour $k > 3$ ([66], théorème 1).

Heuristique d'isolation ([66])

Cette heuristique repose sur le fait qu'une *multiway cut* sépare un terminal des $k - 1$ autres. Ainsi, en prenant les $k - 1$ coupes de poids les plus faibles, on approxime la *multiway cut* minimale correspondante.

Voici les détails de cette heuristique :

1. Pour tout $i \in [1; k]$, trouver une coupe minimale C_i isolant le terminal t_i de tous les autres en traitant ces derniers comme les sommets d'une composante connexe dont on voudrait calculer une coupe minimale relative (voir figure 5.10 illustrant la première méthode décrite dans la section 5.5).
2. Déterminer $i_{\max} \in [1; k]$ tel que $C_{i_{\max}}$ soit la coupe de poids le plus élevé.
3. Renvoyer la coupe C qui est l'union des coupes C_i à l'exception de $C_{i_{\max}}$.

Ainsi on obtient l'ensemble d'arêtes $C \subseteq A$ suivant :

$$C = \bigcup_{i \in [1,k] \setminus i_{\max}} C_i \tag{5.19}$$

avec

$$i_{\max} = \arg\max_{i \in [1,k]} P(C_i) \tag{5.20}$$

Remarque 5.18.
Il est à noter que le résultat de l'heuristique d'isolation n'est pas nécessairement une multiway cut.

Une illustration des étapes de l'heuristique d'isolation et de cette remarque se trouve dans la figure 5.10 où l'on observe que le graphe $(S, A \setminus C)$ de la figure 5.10b possède une composante connexe ne contenant aucun terminal.

Nous verrons dans le chapitre 6 qu'il existe des algorithmes plus génériques dont certains cas particuliers permettent d'approximer une *multiway cut* de poids minimal et qui garantit que l'ensemble d'arête obtenu soit bien une *multiway cut*.

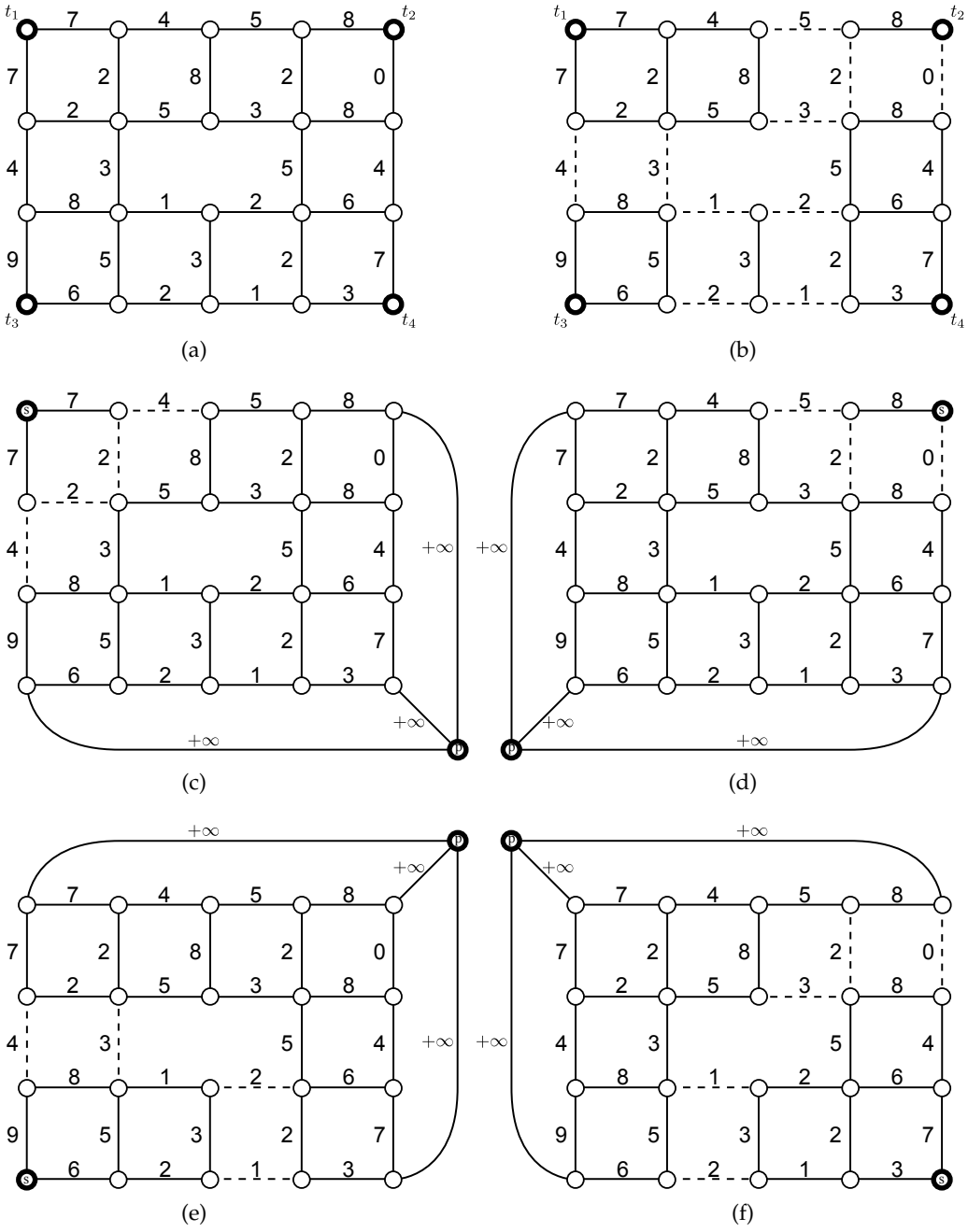


FIGURE 5.10 – (a) Graphe non-orienté G avec, en gras, quatre terminaux t_1, t_2, t_3 et t_4. (b) Graphe G avec, en gras, les terminaux t_1, t_2, t_3 et t_4 et, en pointillés, l'ensemble d'arêtes $C = C_2 \cup C_3 \cup C_4$ qui n'est pas une *multiway cut* (voir (c), (d), (e) et (f)). (c) Graphe G_1 avec, en gras, source/puits et, en pointillés, une coupe minimale C_1 avec $P(C_1) = 12$. (d) Graphe G_2 avec, en gras, source/puits et, en pointillés, une coupe minimale C_2 avec $P(C_2) = 7$. (e) Graphe G_3 avec, en gras, source/puits et, en pointillés, une coupe minimale C_3 avec $P(C_3) = 10$. (f) Graphe G_4 avec, en gras, source/puits et, en pointillés, une coupe minimale C_4 avec $P(C_4) = 8$.

Remarque 5.19.
Dans la suite de ce chapitre, et conformément à la remarque 5.17, dans le cadre de graphes non-orientés, le terme de coupe représentera également la notion de multiway cut. Ainsi, de façon générale, on appellera coupe minimale une coupe de poids minimal relative à un sous-graphe (retombant ainsi sur la définition exacte de multiway cut lorsque ce sous-graphe est constitué de sommets isolés uniquement).
Cette généralisation nous sera notamment utile dans la section 5.7 où l'on mettra en évidence le lien entre coupe minimale et coupe induite par une forêt de poids maximum (ces dernières étant définies comme relatives à un sous-graphe quelconque).

5.7 Coupe minimale et coupe induite par une forêt couvrante de poids maximum

Dans cette section, nous montrons que coupe induite par forêt couvrante de poids maximum et coupe minimale sont liées à travers une modification par une fonction de l'application de poids associée au graphe. Ces résultats, trouvés en collaboration avec Jean-Yves Audibert et Michel Couprie, firent l'objet d'un rapport de recherche [7], d'une publication [8] ainsi que d'un oral à l'*International Symposium on Mathematical Morphology* qui s'est tenu à Rio de Janeiro (Brésil) en Octobre 2007.

Bien que la notion de coupe minimale soit définie dans les graphes orientés, comme annoncé en début de chapitre, nous ne considérerons ici que le cas des graphes non-orientés. En effet, comme expliqué dans les sections 1.2.3 et 5.4, 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 égaux sur un arc et son arc symétrique. C'est pourquoi le lien décrit ici se limite au cadre des graphes non-orientés.

De même, puisque les coupes minimales ne peuvent pas être calculées pour n'importe quelle application de poids, nous considérerons dans cette section une application de poids $P : A \to \mathbb{R}^+$.

D'après la remarque 4.5, nous savons qu'une FCMax pour une application de poids P est aussi une FCMax pour une application de poids $P' = f \circ P$ ou f est une fonction strictement croissante. Ceci est donc également valable pour les coupes par FCMax. Ce n'est cependant pas le cas pour les coupes minimales comme le montre la figure 5.11.

C'est à travers ce phénomène et une fonction correctement choisie que nous montrons qu'il est possible de faire coïncider une coupe minimale avec une coupe par FCMax en modifiant l'application de poids du graphe.

Il est à préciser que ce type de modification de poids diffère de celui présenté dans la remarque 5.9 puisque la modification du poids de chaque coupe en résultant n'est pas le même, d'où le fait qu'une coupe minimale dans la pondération initiale ne le soit plus nécessairement après altération des poids.

On note $P^{[\alpha]}$, et on appelle P puissance α, l'application de poids $P^{[\alpha]} : A \to \mathbb{R}^+$ définie par, pour toute arête $a \in A$, $P^{[\alpha]}(a) = [P(a)]^\alpha$.

Nous utilisons cette nouvelle fonction de poids afin de mettre en évidence le lien qui existe entre coupe minimale et coupe par FCMax dans le théorème suivant.

Théorème 5.20.
Si M est un sous-graphe de G, alors il existe un nombre réel $\beta \in \mathbb{R}^+$ tel que, pour tout $\alpha \geq \beta$, toute coupe minimale relative à M pour $P^{[\alpha]}$ est une coupe par FCMax relative à M pour $P^{[\alpha]}$.

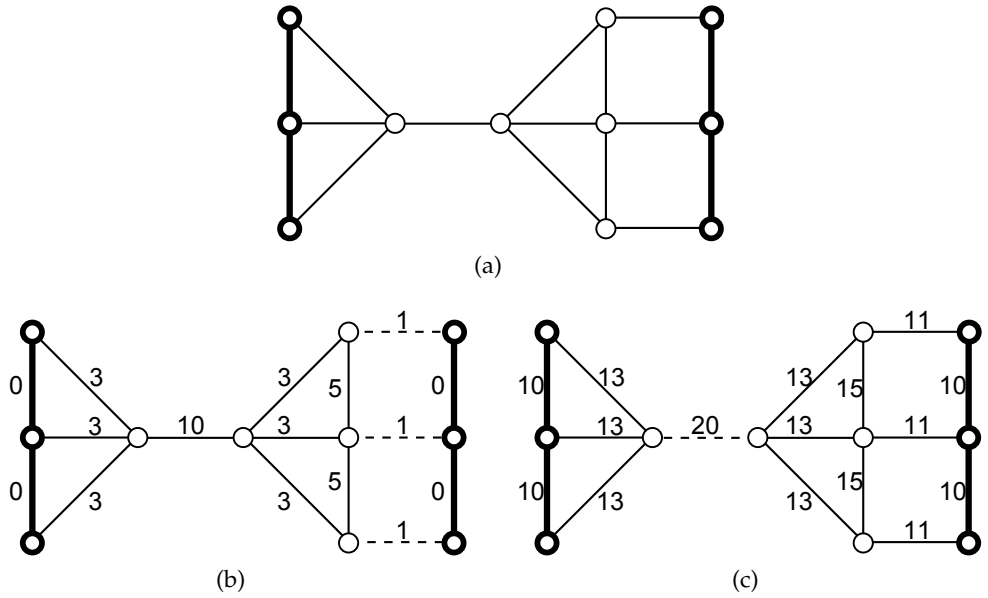


FIGURE 5.11 – Graphe G avec : (a) en gras, un sous-graphe M ; (b) une application de poids P avec, en gras, le sous-graphe M et, en pointillés, la coupe minimale relative à M pour P ; (c) une application de poids $P' = f \circ P$ tel que $f : x \mapsto (x + 10)$ avec, en gras, le sous-graphe M et, en pointillés, la coupe minimale relative à M pour P'.

La preuve de ce théorème se trouve ci-après, en fin de section.

Une illustration de ce théorème se trouve dans la figure 5.12.

Remarque 5.21.
Il est à noter que la réciproque du théorème 5.20 n'est, en général, pas vraie.
Un contre-exemple se trouve dans la figure 5.13 où la coupe par FCMax relative à M pour P de la figure 5.13b n'est pas une coupe minimale relative à M pour $P^{[\alpha]}$, quel que soit $\alpha \in \mathbb{R}+$. Par contre, toute coupe minimale relative à M est une coupe par FCMax relative à M.

D'après la remarque 4.5, nous savons qu'une coupe par FCMax relative à M pour $P^{[\alpha]}$ est également une coupe par FCMax relative à M pour P et réciproquement puisque le changement d'application de poids se fait au travers d'une fonction strictement croissante. Par conséquent, on peut déduire du théorème 5.20 que les coupes par FCMax sont des cas particuliers des coupes minimales.

Soit un ensemble d'arêtes $X \subseteq A$.
On note $P^{[\alpha]}(X)$ la somme des poids à la puissance α des arêtes de X (i.e. $P^{[\alpha]}(X) = \sum_{x \in X} [P(x)]^{\alpha}$).
Cette notation n'est pas sans rappeler celle d'une norme α :

$$\|(x_1, \ldots, x_k)\|_\alpha = \left(|x_1|^\alpha + \ldots + |x_k|^\alpha\right)^{\frac{1}{\alpha}} \tag{5.21}$$

Or, quand α augmente et tend vers l'infini, nous avons :

$$\begin{aligned} \|(x_1, \ldots, x_k)\|_\infty &= \lim_{n \to +\infty} \|(x_1, \ldots, x_k)\|_n \\ &= \max\{|x_1|, \ldots, |x_k|\} \end{aligned} \tag{5.22}$$

Il en découle :

$$\lim_{\alpha \to +\infty} P^{[\alpha]}(X) = \max_{x \in X} \{P^{[\alpha]}(x)\} \tag{5.23}$$

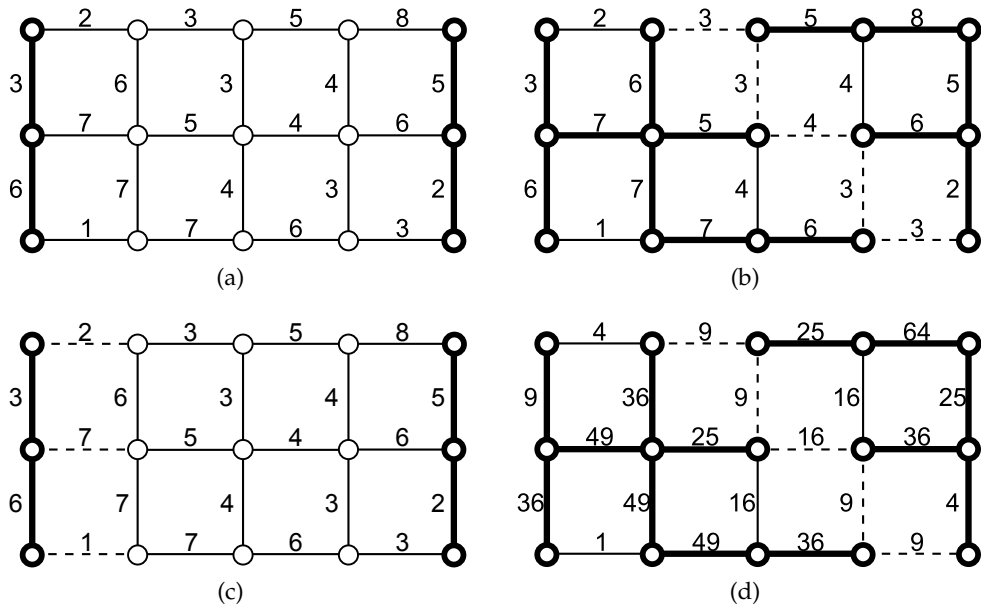


FIGURE 5.12 – Graphe G et application de poids P avec : (a) en gras, un sous-graphe M ; (b) en gras, une FCMax relative à M pour P et, en pointillés, sa coupe induite ; (c) en gras, le sous-graphe M et, en pointillés, une coupe minimale relative à M pour P ; (d) en gras, une FCMax relative à M pour $P^{[2]}$ et, en pointillés, sa coupe induite qui est aussi une coupe minimale relative à M pour $P^{[2]}$.

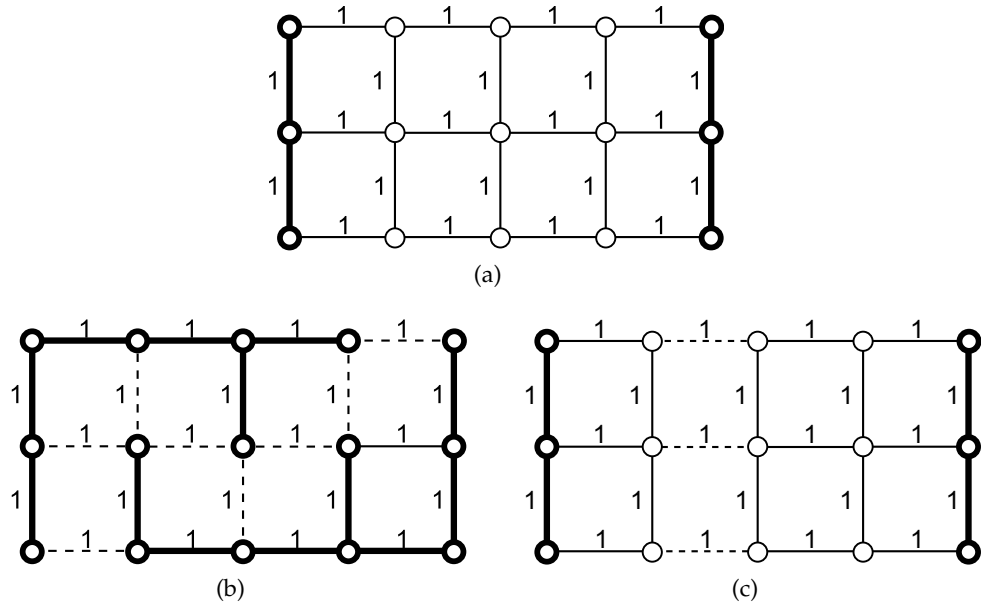


FIGURE 5.13 – Graphe G et application de poids P avec : (a) en gras, un sous-graphe M ; (b) un gras, une FCMax relative à M pour P et, en pointillés, sa coupe induite, qui n'est pas une coupe minimale ; (c) en gras, le sous-graphe M et, en pointillés, une coupe minimale relative à M pour P.

Ainsi, en augmentant les puissances des poids des arêtes du graphe, le poids d'une coupe tend vers le poids de son arête de poids le plus grand. Dans ces conditions, si on considère une coupe minimale, on aura donc une coupe dont l'arête de poids maximal est le plus faible possible, ce qui correspond exactement à une coupe par FCMax.

Remarque 5.22.
Une coupe par FCMax correspond donc à une minimisation globale au sens de la norme infinie tandis qu'une coupe minimale correspond à une minimisation globale au sens de la norme 1.

A la vue de ce qui a été mentionné ci-avant, on déduit qu'une interprétation intuitive de ce résultat est de considérer les coupes par FCMax comme une heuristique gloutonne pour obtenir une coupe minimale ou, tout du moins, une approximation. L'efficacité de cette heuristique est d'autant meilleure que l'écart entre les poids d'origine des arêtes du graphe est grand. En effet, le passage de l'application de poids P à l'application de poids $P^{[\alpha]}$ n'est là que pour faire croître "artificiellement" cet écart.

La figure 5.14 illustre le lien existant entre coupe par FCMax et coupe minimale décrit dans le théorème5.20 dans le cadre de la segmentation d'une image couleur. On peut ainsi y observer l'évolution de la coupe minimale à travers l'augmentation de la puissance des poids des arêtes, représentée par la valeur α. Cette valeur peut ainsi être considérée comme un terme de lissage de la coupe minimale obtenue. En effet, quand α décroît des coupes plus "courtes" (contenant moins d'arêtes) sont obtenues alors que, quand α augmente on trouve des coupes plus "longues" (contenant plus d'arêtes) jusqu'à tomber sur la coupe par FCMax. Une coupe ainsi "relâchée" permet de mieux suivre certains détails du contour d'un objet mais peut tout aussi bien se trouver détournée par du bruit ou des parasites de l'image. Ainsi, ce relâchement n'est pas toujours souhaitable.

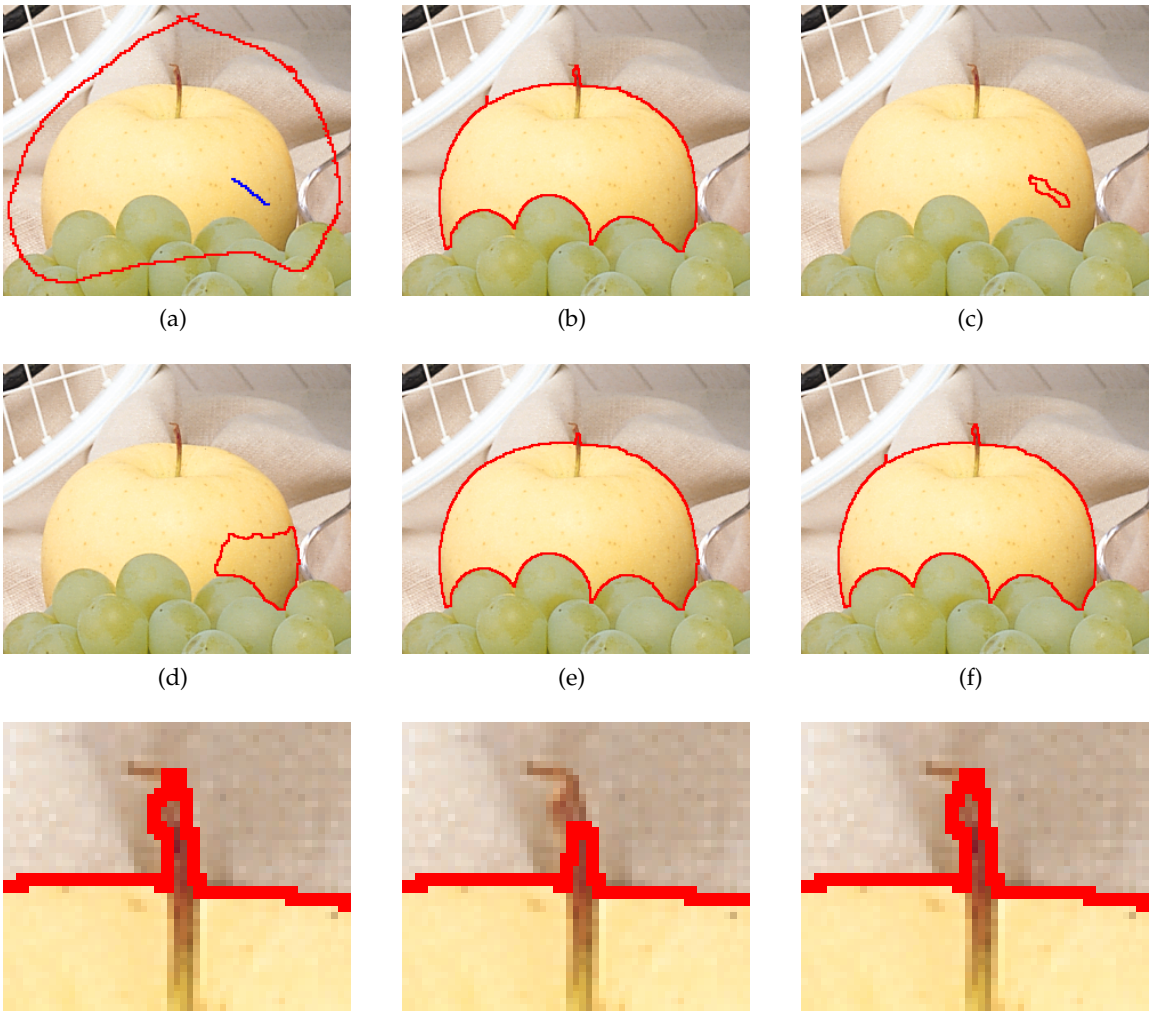
Il est cependant à noter que les coupes minimales, utilisées en segmentation d'image, n'offrent que rarement des résultats aussi "catastrophiques" que celui de la figure 5.14c, comme le montre la figure 5.15.

Il est à noter que, puisque nous savons, d'après la remarque 4.5 et le théorème 4.19, qu'une coupe par FCMax peut-être considérée comme une LPE, à l'application d'une fonction strictement décroissante sur les poids près, nous pouvons déduire du théorème 5.20 que, toujours à cette fonction strictement décroissante près, les LPE sont des cas particuliers des coupes minimales.

Preuve du théorème 5.20

Dans cette preuve, afin d'alléger son écriture, les extensions et forêts mentionnées seront toutes considérées comme relatives au sous-graphe M de G. Nous introduisons ainsi les notations commodes qui suivent.

Notation :
Soit un ensemble d'arêtes A' tel que $A(M) \subseteq A' \subseteq A$.
- une A'**-extension** est une extension relative à M sur (S, A') ;
- une A'**-extension couvrante** est une extension couvrante relative à M sur (S, A') ;
- une A'**-extension maximale** est une extension maximale relative à M sur (S, A') ;
- une A'**-forêt couvrante** est une forêt couvrante relative à M sur (S, A') ;
- une A'**-FCMax** est une FCMax relative à M sur (S, A').



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

FIGURE 5.14 – Segmentation d'une image couleur avec : (a) marqueurs superposés à l'image originale ; (b) coupe par FCMax pour P ; (c) coupe minimale pour P ; (d) coupe minimale pour $P^{[1.4]}$; (e) coupe minimale pour $P^{[2]}$; (f) coupe minimale pour $P^{[3]}$; (g) zoom de l'image (b) ; (h) zoom de l'image (e) ; (i) zoom de l'image (f).

(a) (b) (c)

FIGURE 5.15 – Segmentation d'images couleur avec : (a) marqueurs superposés à l'image originale ; (b) coupe par FCMax pour P ; (c) coupe minimale pour P.

Soit $r = \min\left\{\frac{P(a_i)}{P(a_j)} \text{ tel que } (a_i; a_j) \in A^2 \text{ avec } P(a_i) > P(a_j)\right\}$ et prenons

$$\beta = \frac{\log |A|}{\log r}. \tag{5.24}$$

Soit C une coupe minimale relative à M pour $P^{[\alpha]}$ avec $\alpha \geq \beta$.
Soit F une A-FCMax.
Soit $\tilde{F}$ une $\overline{C}$-FCMax (avec, pour rappel, $\overline{C} = A \setminus C$).

Pour prouver le résultat du théorème 5.20, autrement dit que $\tilde{F}$ est une A-FCMax, nous allons montrer que :

(i) $P(\tilde{F}) = P(F)$,

(ii) $\tilde{F}$ est une A-forêt couvrante.

Pour obtenir (i), nous prouvons que F et $\tilde{F}$ ont le même nombre d'arêtes de chaque type de poids.

Soient $\mathcal{P} = \langle p_1, \ldots, p_\ell \rangle$ les différents poids des arêtes de G, tels que :

$$\forall i \in [1; \ell], \exists a \in A \mid P(a) = p_i, \tag{5.25}$$
$$\forall a \in A, \exists i \in [1; \ell] \mid P(a) = p_i \tag{5.26}$$

et

$$\forall i, j \in [1; \ell], \text{ si } i \neq j \text{ alors } p_i \neq p_j, \tag{5.27}$$

présentés par ordre décroissant avec

$$p_1 > \cdots > p_\ell. \tag{5.28}$$

On appelle $\mathcal{P}$ la **séquence induite** de G pour P. Cette séquence induite permet ainsi de représenter les valeurs des poids de P une unique fois en les classant par ordre décroissant.

Soit $n_h(X)$ le nombre d'arêtes de poids p_h du sous-graphe X de G. Ainsi, on a évidemment :

$$|A(X)| = \sum_{h=1}^{\ell} n_h(X), \tag{5.29}$$

Le poids de X peut alors être exprimé comme suit :

$$P(X) = \sum_{h=1}^{\ell} n_h(X) p_h. \tag{5.30}$$

Pour avoir (i), il suffit de montrer que pour tout $h \in [1; \ell]$:

$$n_h(F) = n_h(\tilde{F}). \tag{5.31}$$

Nous définissons

$$A^{(h)} = \{a \in A; P(a) \geq p_h\} \cup A(M) \tag{5.32}$$

et introduisons

$$F^{(h)} = \left(V, A(F) \cap A^{(h)}\right), \tag{5.33}$$
$$\tilde{F}^{(h)} = \left(V, A(\tilde{F}) \cap A^{(h)}\right). \tag{5.34}$$

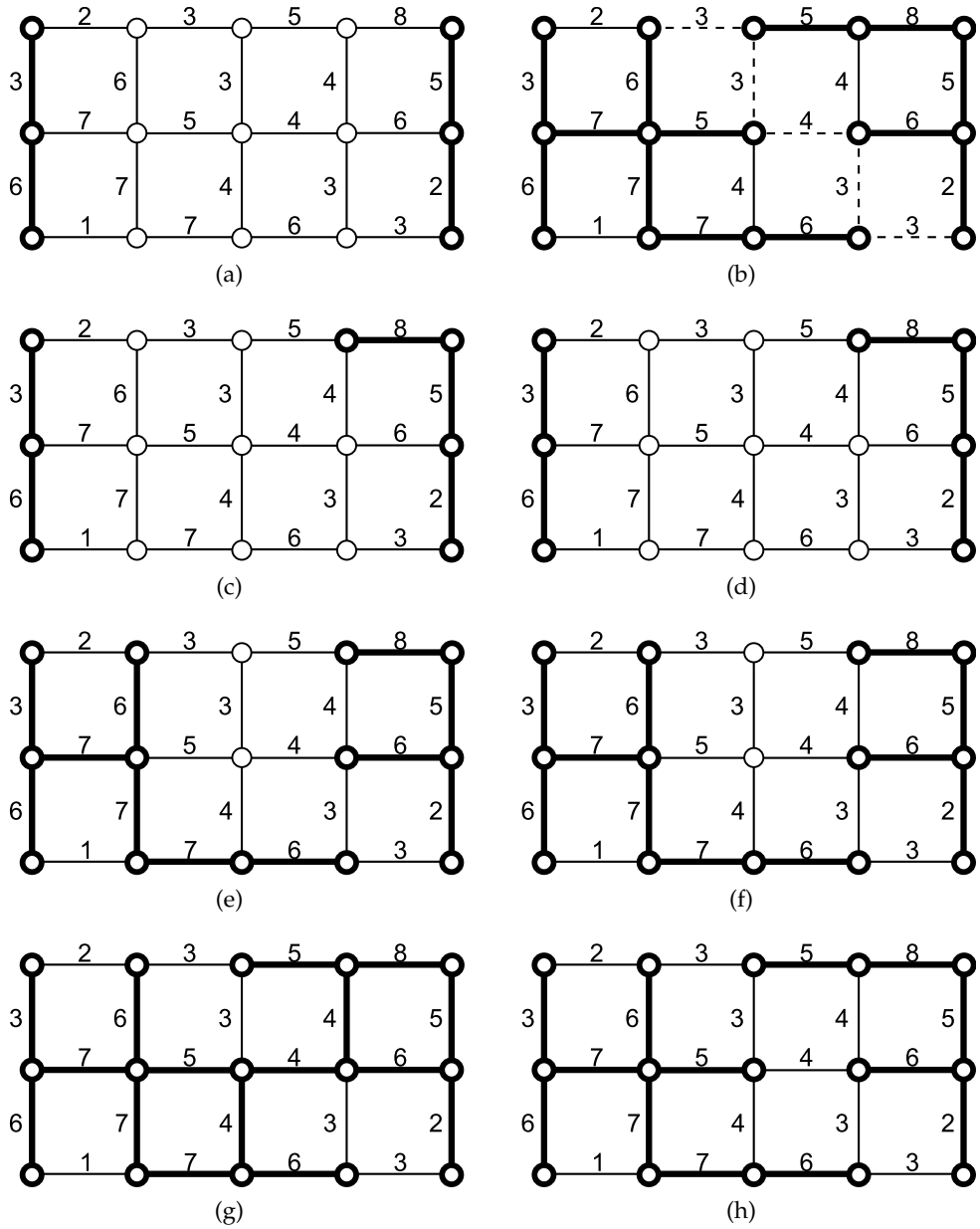


FIGURE 5.16 – Graphe G et application de poids P (dont la séquence induite est $\langle 9,8,7,6,5,4,3,2,1\rangle$) avec (en gras) : (a) un sous-graphe M ; (b) une FCMax relative à M pour P ; (c) $G^{(2)}$ ($p_2 = 8$) ; (d) $F^{(2)}$; (e) $G^{(4)}$ ($p_4 = 6$) ; (f) $F^{(4)}$; (g) $G^{(6)}$ ($p_6 = 4$) ; (h) $F^{(6)}$.

Une illustration de ces notations se trouve dans la figure 5.16. Celles-ci ne sont pas sans rappeler la définition du seuillage supérieur présentée dans la section 1.4.9.3.

Il est à noter que $F = F^{(\ell)}$ et $\tilde{F} = \tilde{F}^{(\ell)}$. Pour obtenir l'équation 5.31, nous allons prouver par récurrence sur k que, pour tout $1 \leq h \leq k$:

$$n_h(F^{(k)}) = n_h(\tilde{F}^{(k)}). \tag{5.35}$$

La preuve de l'équation 5.35 requiert les trois lemmes suivants.

Lemme 5.23.
Pour tout $h \in [1;\ell]$, $F^{(h)}$ est une $A^{(h)}$-forêt couvrante.

Preuve du lemme 5.23.
Soit $G^{(h)} = \left(S, A^{(h)}\right)$. Nous commençons avec deux affirmations évidentes :
- Toute composante connexe de M est incluse dans une composante connexe $K_{F^{(h)}}$ de $F^{(h)}$, puisque, par définition de $F^{(h)}$, nous avons $A(M) \subseteq F^{(h)}$.
- Le graphe connexe $K_{F^{(h)}}$ ne peut contenir plus d'une composante connexe de M puisque, autrement, une des composantes connexes de F contiendrait plus d'une composante connexe de M, ce qui est impossible.

Ainsi, pour prouver que $F^{(h)}$ est une $A^{(h)}$-extension, il suffit de vérifier que pour toute composante connexe $K_{G^{(h)}}$ de $G^{(h)}$ (que l'ensemble de sommets $S(K_{G^{(h)}}) \cap S(M)$ soit vide ou non), le graphe $K_{G^h} \cap F^h$ est connexe. Nous allons le prouver par l'absurde. Supposons que le graphe $K_{G^h} \cap F^h$ n'est pas connexe. Alors il existe une arête $a = \{x, y\} \in A(K_{G^{(h)}})$ telle que $a \notin A(F^{(h)})$. Puisque $P(a) \geq p_h$, alors nous avons également $a \notin A(F)$. Il y a alors soit une chaîne élémentaire reliant x et y dans F contenant une arête $a' \in A \setminus A^{(h)}$ ou une chaîne élémentaire reliant x et un sommet de M dans F contenant une arête $a' \in A \setminus A^{(h)}$. Dans les deux cas, si l'on retire a' et que l'on ajoute a à F, nous obtenons une A-forêt couvrante dont le poids est plus grand que celui de F (puisque $P(a) \geq p_h > P(a')$). Ceci contredit le fait que F est une A-forêt couvrante de poids maximum. Ainsi, $F^{(h)}$ est une $A^{(h)}$-extension.
Maintenant, considérons l'arête $a'' \in A\left(F^{(h)}\right)$. Le sous-graphe $\left(S, A(F^{(h)}) \setminus \{a''\}\right)$ n'est pas une $A^{(h)}$-extension puisque l'on peut vérifier qu'autrement $\left(S, A(F) \setminus \{a''\}\right)$ serait une A-extension, ce qui contredit le fait que F est une A-forêt couvrante. Ainsi, $F^{(h)}$ est une $A^{(h)}$-forêt couvrante. □

Remarque 5.24.
En utilisant le même type de construction que celle utilisée pour prouver que $F^{(h)}$ est une $A^{(h)}$-extension, il est possible de prouver que $F^{(h)}$ est en fait une $A^{(h)}$-FCMax, mais cette propriété ce sera pas utilisée ici.

Lemme 5.25.
Pour tout $h \in [1;\ell]$, $\tilde{F}^{(h)}$ est une $A^{(h)}$-forêt couvrante.

Preuve du lemme 5.25.
Soient $\tilde{A}^{(h)} = A^{(h)} \setminus C$, $\tilde{G}^{(h)} = \left(S, \tilde{A}^{(h)}\right)$ et $G^{(h)} = \left(S, A^{(h)}\right)$. Nous allons prouver que $\tilde{G}^{(h)}$ est une $A^{(h)}$-extension. Nous commençons en notant que :
- toute composante connexe de M est incluse dans une composante connexe $K_{\tilde{G}^{(h)}}$ de $\tilde{G}^{(h)}$, puisque la coupe C relative à M sur G ne contient aucune arête de $A(M)$,

– le graphe connexe $K_{\tilde{G}^{(h)}}$ ne peut contenir plus d'une composante connexe de M puisque, autrement, une des composantes connexes de $\tilde{G}^{(\ell)} = (S, A \setminus C)$ contiendrait plus d'une composante connexe de M, ce qui est impossible par définition de C.

Ainsi, pour prouver que $\tilde{G}^{(h)}$ est une $A^{(h)}$-extension, il suffit de vérifier que, pour toute composante connexe $K_{G^{(h)}}$ de $G^{(h)}$ (que l'ensemble de sommets $S(K_{G^{(h)}}) \cap S(M)$ soit vide ou non), le graphe $K_{G^{(h)}} \cap \tilde{G}^{(h)}$ est connexe. Nous allons le prouver par l'absurde. Supposons que le graphe $K_{G^{(h)}} \cap \tilde{G}^{(h)}$ n'est pas connexe. Alors, il existe une arête $a = \{x, y\} \in A(K_{G^{(h)}})$ telle que $a \in C$. Autrement dit, a est dans la coupe minimale C relative à M sur G. Puisque les sommets x et y sont tous deux dans $S(M)$, en partant de C, on peut construire une coupe C' relative à M sur G en enlevant l'arête a et en ajoutant des arêtes de $A \setminus A^{(h)}$ de façon à s'assurer que C' soit bien une coupe. Pour $P^{[\alpha]}$, le poids total des arêtes rajoutées à $C \setminus \{a\}$ est plus petit que $P^{[\alpha]}(a)$, puisque, d'après l'équation 5.24 et $\alpha \geq \beta$, nous avons

$$\begin{aligned} \alpha &\geq \frac{\log |A|}{\log \frac{p_h}{p_{h+1}}} \\ \log \frac{p_h^\alpha}{p_{h+1}^\alpha} &\geq \log |A| \\ p_h^\alpha &\geq |A| p_{h+1}^\alpha \end{aligned} \tag{5.36}$$

et, par conséquent,

$$P^{[\alpha]}(a) = [P(a)]^\alpha \geq p_h^\alpha \geq |A| p_{h+1}^\alpha > \sum_{i=h+1}^{\ell} n_i(G) p_i^\alpha. \tag{5.37}$$

Ainsi, la coupe C' serait telle que $P(C') < P(C)$, ce qui est impossible. C'est pourquoi $\tilde{G}^{(h)}$ est une $A^{(h)}$-extension.

Nous allons maintenant prouver par l'absurde que $\tilde{G}^{(h)}$ est également une $A^{(h)}$-extension maximale. Supposons qu'il existe une arête a que l'on peut ajouter à $\tilde{G}^{(h)}$ de sorte à ce que $\check{G}^{(h)} = \big(S, A(\tilde{G}^{(h)}) \cup \{a\}\big)$ soit aussi une $A^{(h)}$-extension. Soit $\check{C}^{(h)} = A\big(G^{(h)}\big) \setminus A\big(\check{G}^{(h)}\big)$. Alors, en partant de $\check{C}^{(h)}$, on peut construire une coupe C' relative à M sur G en ajoutant des arêtes de $A \setminus A^{(h)}$de façon à s'assurer que C' est une coupe. En utilisant le même argument que dans l'équation 5.37, on peut vérifier que cela mènerait à avoir $P(C') < P(C)$, ce qui est impossible. Ainsi, $\tilde{G}^{(h)}$ est à la fois une extension maximale relative à M sur $G^{(h)}$ et une extension maximale relative à M sur $\tilde{G}^{(h)}$.

Puisqu'une extension maximale est une extension couvrante, on déduit de la remarque 4.2 que $\tilde{F}^{(h)}$ est une $A^{(h)}$-forêt couvrante. □

Remarque 5.26.
Nous avons, comme résultat dérivé de cette preuve, que $A^{(h)} \cap C$ est une coupe minimale relative à M sur $\big(S, A^{(h)}\big)$.

Lemme 5.27.
Soit $A(M) \subseteq A' \subseteq A$.
Toute A'-forêt couvrante possède le même nombre d'arêtes.

Preuve du lemme 5.27.
Il suffit de vérifier que le nombre d'arêtes d'une A'-forêt couvrante F' est exactement

$$|A(F')| = |A(M)| + \sum_K \big\{ |S(K)| - |\mathcal{C}(K \cap M)| - \max\big(|S(K \cap M)| - 1, 0\big) \big\}, \tag{5.38}$$

où la somme est faite sur les composantes connexes de (S, A').
Nous rappelons que $|\mathcal{C}(X)|$ représente le nombre de composantes connexes de X. □

Maintenant que nos trois lemmes ont été prouvés, nous pouvons retourner à la preuve de l'équation 5.35 qui suit.

Preuve par récurrence sur k de l'équation 5.35.

- Pour $k = 1$: d'après les lemmes 5.23, 5.25 et 5.27, $F^{(1)}$ et $\tilde{F}^{(1)}$ possèdent le même nombre d'arêtes. Puisque, dans les deux sous-graphes, les seules arêtes de poids inférieur à p_1 sont exactement les arêtes de M de poids inférieur à p_1, nous obtenons $n_1\big(F^{(1)}\big) = n_1\big(\tilde{F}^{(1)}\big)$.
- Considérons maintenant que l'équation 5.35 est valable pour k. Montrons alors qu'elle est également valable pour $k+1$. Pour tout $1 \leq h \leq k$, nous avons

$$n_h\big(F^{(k+1)}\big) = n_h\big(F^{(k)}\big) = n_h\big(\tilde{F}^{(k)}\big) = n_h\big(\tilde{F}^{(k+1)}\big), \tag{5.39}$$

  où la seconde égalité est obtenue à partir de l'équation 5.35 pour k. Maintenant, à partir des lemmes 5.23, 5.25 et 5.27, $F^{(k+1)}$ et $\tilde{F}^{(k+1)}$ possèdent le même nombre d'arêtes. Puisque dans les deux sous-graphes, les seules arêtes de poids inférieur à p_{k+1} sont exactement les arêtes de M de poids inférieur à p_{k+1}, nous obtenons $n_{k+1}\big(F^{(k+1)}\big) = n_{k+1}\big(\tilde{F}^{(k+1)}\big)$, ce qui finit la récurrence. □

Ainsi, la preuve de (i) est terminée. Pour (ii), il suffit de remarquer que, puisque nous avons $A = A^{(\ell)}$ et $F = F^{(\ell)}$, c'est une conséquence directe du lemme 5.25.

Remarque 5.28.
Dans la preuve du théorème 5.20, le fait que α soit suffisamment grand est nécessaire uniquement pour s'assurer que pour tout $h \in [1; \ell - 1]$, $G^{(h)} = \big(S, A^{(h)} \setminus C\big)$ est une $A^{(h)}$-extension.
En traitement d'image, il arrive fréquemment que la coupe minimale relative à $P^{[\gamma]}$ avec une valeur de γ petite (typiquement $\gamma = 1$) "lisse" trop le contour ainsi obtenu. C'est pourquoi, en essayant des valeurs de γ plus élevées, il est possible d'"affiner" le contours.
Il est à noter que cette remarque peut ainsi permettre d'accélérer le calcul d'une coupe minimale pour $P^{[\gamma']}$ avec $\gamma' > \gamma$ quand la coupe minimale pour $P^{[\gamma]}$ est déjà connue. En effet, le flot maximal obtenu pour $P^{[\gamma]}$ peut servir d'initialisation au calcul de celui pour $P^{[\gamma']}$ avec la méthode des dynamic cuts, présentée dans la sous-section 5.3.3.1, mais comme tous les poids sont modifiés, l'efficacité de cette méthode (linéaire en fonction du nombre de poids modifiés) ne sera sans doute pas optimale. On pourra alors lui préférer la méthode des active cuts, présentée dans la sous-section 5.3.3.2.

5.8 Récapitulatif et conclusion

Le lien entre coupe minimale et coupe par FCMax présenté dans ce chapitre vient compléter les différents liens présentés dans le chapitre 4 et met en évidence deux choses :

- une coupe par FCMax peut être considérée comme une heuristique gloutonne d'approximation d'une coupe minimale d'autant plus proche que les écarts entre les poids des arêtes du graphe sont grands et ayant l'avantage d'avoir des algorithmes quasi-linéaires ;
- une coupe par FCMax correspond à une minimisation globale au sens de la norme infinie.

Ce nouveau lien, cumulé à ceux présentés dans le chapitre 4, nous permet de nous rendre compte que coupe par FCMin, coupe par FCMax, coupe par FCCMA, coupe minimale et LPE sont toutes liées par certains critères ou bien complètement équivalentes comme la coupe par FCCMA et la LPE.

Bien que de définitions complètement différentes, nous pouvons ainsi noter que LPE et coupe minimale sont liées au travers de leurs relations propres avec les forêts couvrantes de poids extremum et le choix d'une pondération adaptée. Ainsi, l'augmentation de la puissance des poids d'un graphe va faire converger toute coupe minimale vers une coupe par FCMax, qui peut donc être assimilée à une LPE. La LPE devenant alors un cas particulier de coupe minimale.

Il est néanmoins à noter que calculer une LPE par ce biais serait une perte de temps étant donné la complexité plus élevée du calcul et que, excepté dans le cas de marqueurs composés de deux composantes connexes uniquement, le résultat obtenu n'est, généralement, qu'une approximation. Au contraire, comme nous l'avons vu, une LPE peut être calculée en temps linéaire ou quasi-linéaire (selon qu'elle est relative aux minima régionaux ou non). Malheureusement, aucun lien "inverse" permettant de trouver une coupe minimale à partir d'un algorithme (quasi-)linéaire de LPE ne semble envisageable, mais une LPE peut sans doute être considérée comme une coupe d'initialisation intéressante dans la méthode des *active cuts* (voir section 5.3.3.2).

Malgré tout, il est à noter que la puissance de la pondération d'un graphe peut ainsi être utilisée comme paramètre de lissage des coupes minimales, partant ainsi de la coupe minimale sur les poids d'origines, celle-ci se modifiant de sorte à épouser davantage de détails du contour au fur et à mesure que la puissance augmente sans pour autant être aussi sensible aux défauts de la LPE tel que le phénomène de fuite (présenté dans la sous-section 3.6.1), comme illustré dans la figure 5.14. Ce paramètre, intrinsèque à la coupe minimale, pourrait ainsi offrir de nouvelles possibilités d'utilisation des coupes minimales. Ceci doit cependant être modéré par le fait que, dans certains cas, le résultat de la coupe minimale sur les poids d'origine peut être considéré comme le meilleur (voir la figure 5.15).

Enfin, il est à préciser que la valeur de β du théorème 5.20 que nous fournissons dans la preuve (voir en fin de section 5.7) comme suffisante peut s'avérer être un nombre extrêmement grand alors que, expérimentalement, une valeur plus petite peut souvent suffire en pratique. Trouver une borne plus basse pour β demeure un problème ouvert.

Deuxième partie

Graphes et applications en traitement d'images

Problème d'étiquetage et minimisation d'énergie

6

Beaucoup de problèmes en vision par ordinateur ou en traitement d'images peuvent se ramener à un problème d'étiquetage, autrement dit, comment attribuer la bonne étiquette à chaque élément du problème concerné.

Nous débutons ce chapitre en définissant ce qu'est un problème d'étiquetage, puis nous rappelons comment celui-ci peut se résoudre dans un cadre probabiliste avec les champs aléatoires de Markov avant de le ramener à un problème de minimisation d'énergie.

La seconde partie de ce chapitre est consacrée aux méthodes de résolution de tels problèmes reposant sur les coupes minimales de graphes. Nous commençons en traitant le cas d'un problème à seulement deux étiquettes et montrons comment il a pu être étendu pour obtenir un résultat approché dans le cas général ou bien encore un résultat optimal sous certaines conditions. Une de ces méthodes est, par la suite, utilisée dans les deux applications présentées dans les chapitres 7 et 8.

Les méthodes de résolution par $\{\alpha, \beta\}$-*swaps* ou par α-*expansions* présentées dans la section 6.2.2 ont été légèrement modifiées par rapport à leurs versions d'origines parues dans [35, 36, 37, 38]. Ces modifications, mises en évidence dans les remarques 6.9 et 6.11 ont pour but principal d'accélérer leur exécution en réduisant le nombre de sommets et d'arcs sur lequel doivent être calculés la coupe minimale.

Enfin, nous ferons un tour d'horizon d'autres méthodes de minimisation d'énergie parmi les plus couramment utilisées.

Sommaire du chapitre

Figures du chapitre

6.1 Problème d'étiquetage et minimisation d'énergie

6.1.1 Problème d'étiquetage

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** $\mathcal{P}$ 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 p de $\mathcal{P}$ une unique étiquette $\ell = x_p$ de $\mathcal{L}_p$ qui est le sous-ensemble d'étiquettes de $\mathcal{L}$ qui peuvent être prises pour le site p. On forme ainsi une partition des sites $P = \{\mathcal{P}_\ell \mid \ell \in \mathcal{L}\}$ où $\mathcal{P}_\ell = \{p \in \mathcal{P} \mid x_p = \ell\}$ est une région de la partition.

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

Soit $\mathcal{X}$ 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 $|\mathcal{X}| = n^m$. 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.

6.1.2 Champs aléatoires de Markov et minimisation d'énergie

Chaque solution possible d'un problème d'étiquetage doit être estimée selon des critères propres à l'application dans laquelle il intervient afin d'en déterminer une comme "meilleure" que les autres selon les critères de l'application considérée. Pour ce faire, nous introduirons dans cette section les champs aléatoires de Markov et montrerons comment ce cadre probabiliste se révèle propice à cette estimation, la meilleure solution étant alors celle offrant la plus forte probabilité. Nous montrerons ensuite qu'il est possible d'en déduire une reformulation, appelée énergie, sous la forme d'une fonction de $\mathcal{X}$ dans $\mathbb{R}$ dont la valeur la plus faible est donnée par la meilleure solution d'étiquetage.

Davantage de détails peuvent être trouvés dans les livres [45, 138].

Nous redéfinissons ci-dessous notre problème d'étiquetage en termes probabilistes.

On appelle **champ aléatoire**, ou simplement **champ**, un ensemble de variables aléa-

toires $X = (X_1, \ldots, X_m)$ défini sur $\mathcal{P}$ tel que, pour tout $p \in \mathcal{P}$, X_p peut prendre n'importe quelle valeur dans $\mathcal{L}_p$.

On appelle **configuration**, ou **réalisation**, du champ X une combinaison $x = (x_1, \ldots, x_m)$ de $\mathcal{X}$.

Les **champs aléatoires de Markov** (**MRF**, pour *Markov random fields* en anglais) furent introduits en traitement d'images par Stuart et Donald Geman en 1984 [90] afin de définir les interactions entre sites au travers d'un voisinage $\mathcal{V}$ défini sur $\mathcal{P}$. Nous allons donc tout d'abord définir la notion de voisinage dans les champs aléatoires avant de présenter la définition des champs aléatoires de Markov.

Si l'on considère un site $p \in \mathcal{P}$, alors son **voisinage** $\mathcal{V}_p$ est l'ensemble des sites de $\mathcal{P}$ respectant les conditions suivantes :

- $p \notin \mathcal{V}_p$, et
- $q \in \mathcal{V}_p$ si et seulement si $p \in \mathcal{V}_q$.

Le voisinage $\mathcal{V}$ est donc l'ensemble des paires de sites $\{p, q\}$ telles que $q \in \mathcal{V}_p$ (et donc $p \in \mathcal{V}_q$).

Remarque 6.1.
Il est à noter que le couple $(\mathcal{P}, \mathcal{V})$ est un graphe non-orienté.

Une illustration de la remarque 6.1 se trouve dans la figure 6.1.

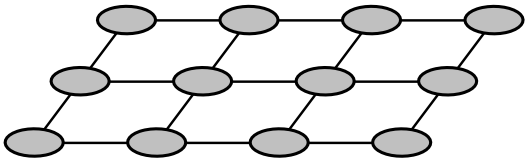


FIGURE 6.1 – Graphe non-orienté $G = (\mathcal{P}, \mathcal{V})$ équivalent à un champ de Markov.

Soit un sous-ensemble de sites $\mathcal{P}' \subseteq \mathcal{P}$.
On note $x_{\mathcal{P}'}$ l'ensemble des variables aléatoires x_p telles que $p \in \mathcal{P}'$.

On appelle **champ aléatoire de Markov**, ou **réseau de Markov**, un champ aléatoire ayant les propriétés suivantes :

- $\forall x \in \mathcal{X}, P(X = x) > 0$, et
- $\forall p \in \mathcal{P}, P(x_p \mid x_{\mathcal{P} \setminus \{p\}}) = P(x_p \mid x_{\mathcal{V}_p})$.

La première condition précise que chaque configuration est possible (cette hypothèse est requise pour assurer que la probabilité jointe est bien définie - voir [27]). La seconde stipule que l'étiquette d'un site p ne dépend que de son voisinage $\mathcal{V}_p$.

Ainsi, les MRF sont un modèle de probabilité conditionnelle tel que la probabilité de l'état observé d'un site ne dépend que de l'état de ses voisins et uniquement de ceux-ci.

Dans un champ aléatoire de Markov, on appelle **clique** un ensemble C de sites (i.e. $C \subseteq \mathcal{P}$) tel que $\forall p, q \in C, p \in \mathcal{V}_q$. Autrement dit, chaque site d'une clique est voisin de tous les autres sites de la clique. On note $\mathcal{C}$ l'ensemble des cliques de $\mathcal{P}$.

Il est à noter que, en considérant les MRF sous la forme de graphes (voir remarque 6.1), cette définition rejoint exactement celle de la sous-section 1.4.6.

On cherche à estimer une réalisation x. Pour cela, nous déterminons la probabilité $P(x \mid d)$ d'avoir une réalisation x du problème d'étiquetage en fonction des données

d'entrée du problème exprimées par la réalisation d. Nous avons alors :

$$\begin{aligned} P(x \mid d) &= \frac{P(x \cap d)}{P(d)} \\ &= \frac{P(d \mid x)P(x)}{P(d)} \end{aligned} \tag{6.1}$$

La recherche de la meilleure solution au problème d'étiquetage se fait grace à une **maximisation a posteriori** (**MAP**) sur un champ aléatoire de Markov (voir [90]). Cela consiste à trouver la réalisation optimale x_{opt} ayant la plus forte probabilité a posteriori $P(x \mid d)$. Autrement dit, la réalisation x_{opt} est telle que :

$$x_{opt} = \arg\max_{x \in \mathcal{X}} \{P(x \mid d)\} \tag{6.2}$$

Puisque l'on cherche à maximiser $P(x \mid d)$ en fonction de x et que le terme $P(d)$ peut être considéré comme une constante, cela revient à chercher la réalisation x_{opt} telle que :

$$x_{opt} = \arg\max_{x \in \mathcal{X}} \{P(d \mid x)P(x)\} \tag{6.3}$$

Le terme $P(x)$ représente la probabilité que l'on puisse avoir la réalisation x comme solution du problème d'après son agencement, autrement dit, d'après les liens existants entre chaque site et ses voisins. Ces liens s'expriment grâce à des fonctions sur les cliques.

Soit, pour toute clique $C \subseteq \mathcal{P}$, la fonction V_C de $\mathcal{X}$ dans $\mathbb{R}$ représentant la cohérence des étiquettes de la clique C, attribuées par la réalisation x, entre elles. Ainsi, la fonction V_C retourne une valeur d'autant plus petite que les étiquettes de la clique C sont cohérentes entre elles.

On appelle **distribution de Gibbs** l'expression de la probabilité :

$$P(X = x) = \frac{1}{Z} \exp\Big(- \sum_{C \in \mathcal{C}} V_C(x) \Big) \tag{6.4}$$

où Z est une constante de normalisation.

On appelle **champ aléatoire de Gibbs** (**GRF**, pour *Gibbs random fields* en anglais) un champ aléatoire vérifiant la distribution de Gibbs.

Le théorème d'Hammerley-Clifford [27, 90] prouve l'équivalence entre GRF et MRF, nous permettant ainsi d'utiliser la formulation de la distribution de Gibbs (équation 6.4) dans le cadre des MRF et donc dans celui de notre problème d'étiquetage. Cependant, par la suite, nous ne considérerons dans la distribution de Gibbs que les fonctions sur des cliques de deux sites, qui sont donc les arêtes du graphe représentant le MRF (voir remarque 6.1). En remplaçant la fonction V_C par, pour toute paire de sites $\{p, q\} \in \mathcal{V}$, la fonction $V_{\{p,q\}}$ de $\mathcal{L}^2$ dans $\mathbb{R}$ spécifique aux cliques de deux sites, on obtient la formulation suivante :

$$P(X = x) = \frac{1}{Z} \exp\Big(- \sum_{\{p,q\} \in \mathcal{V}} V_{\{p,q\}}(x_p, x_q) \Big) \tag{6.5}$$

Il est à noter que, puisque $\{p, q\}$ est une paire de sites (autrement dit un ensemble de sites) et non un couple (qui, lui, serait ordonné), nous avons nécessairement :

$$V_{\{p,q\}}(x_p, x_q) = V_{\{p,q\}}(x_q, x_p) \tag{6.6}$$

La relation entre deux sites voisins est donc **isotrope** (i.e. indépendante de l'orientation).

Le terme $P(d \mid x)$ représente la probabilité que l'on puisse avoir les données observées représentées par d en connaissant l'étiquetage x. Puisque les liens entre sites sont compris dans le terme $P(x)$ du modèle d'étiquetage, les sites peuvent donc être considérés comme indépendants dans le terme $P(d \mid x)$, nous permettant ainsi de l'exprimer comme suit :

$$P(d \mid x) = \prod_{p \in \mathcal{P}} P(d_p \mid x_p) \tag{6.7}$$

Soit, pour tout $p \in \mathcal{P}$, la fonction D_p de $\mathcal{L}$ dans $\mathbb{R}$ représentant la cohérence des données du problème en fonction de l'étiquetage pour le site p. Ainsi, la fonction D_p retourne une valeur d'autant plus petite que l'étiquette est cohérente avec les données.

Nous pouvons alors formuler :

$$P(d_p \mid x_p) = K \exp\big(-D_p(x_p)\big) \tag{6.8}$$

où K est une constante de normalisation.

A la normalisation près, nous avons alors :

$$P(d \mid x) = \exp\Big(-\sum_{p \in \mathcal{P}} D_p(x_p)\Big) \tag{6.9}$$

Le problème de MAP sur champ aléatoire de Markov peut donc, en réintroduisant les équations 6.5 et 6.9, aux constantes de normalisations près, dans l'équation 6.3, revenir à chercher la réalisation x_{opt} telle que :

$$x_{opt} = \arg\max_{x \in \mathcal{X}} \Big\{ \exp\big(-\sum_{p \in \mathcal{P}} D_p(x_p) - \sum_{\{p,q\} \in \mathcal{V}} V_{\{p,q\}}(x_p, x_q)\big) \Big\} \tag{6.10}$$

ce qui est équivalent à :

$$x_{opt} = \arg\min_{x \in \mathcal{X}} \Big\{ \sum_{p \in \mathcal{P}} D_p(x_p) + \sum_{\{p,q\} \in \mathcal{V}} V_{\{p,q\}}(x_p, x_q) \Big\} \tag{6.11}$$

On appelle **énergie** la quantité :

$$E(x) = \underbrace{\sum_{p \in \mathcal{P}} D_p(x_p)}_{\text{termes directs}} + \underbrace{\sum_{\{p,q\} \in \mathcal{V}} V_{\{p,q\}}(x_p, x_q)}_{\text{termes de voisinage}} \tag{6.12}$$

L'énergie $E(x)$ représente donc l'évaluation quantitative que l'on fait de la réalisation x. Bien sûr, une telle énergie est entièrement dépendante du problème à résoudre au travers du choix de :

- la fonction D_p, qui représente un terme directement dépendant des données du site p concerné, aussi appelé attache aux données, incitant l'étiquette d'un site à être calculée en fonction des données du problème et de manière indépendante par rapport aux autres sites ;
- la fonction $V_{\{p,q\}}$, qui représente un terme dépendant des données des sites p et q incitant les étiquettes des différents sites à être calculées les unes en fonction des autres.

Remarque 6.2.
Il est à noter que si l'on modifie une énergie $E(x)$ pour obtenir une énergie $E'(x) = E(x) + K$ où K est une constante, alors une configuration x^* minimise l'énergie E si et seulement si x^* minimise l'énergie E'.

La minimisation d'une énergie du type de l'équation 6.12 est donc une méthode de résolution d'un problème d'étiquetage. Néanmoins, dans le cas général, cela s'avère être NP-complet. Nous verrons dans la section 6.2 le moyen d'approximer cette minimisation dans le cas général à l'aide de coupes minimales. Une présentation succinte d'autres méthodes d'approximation (non utilisées dans la suite de ce mémoire) sera faite dans la section 6.3.

6.1.3 Energies particulières

Nous présentons brièvement ci-après deux types d'énergies particulières couramment utilisées en vision par ordinateur.

6.1.3.1 Préservation de discontinuité

Dans certains cas, il est important que le changement d'étiquette ne soit pas trop pénalisant afin que deux sites voisins puissent avoir des étiquettes "très" différentes.

De façon informelle, une énergie **préserve la discontinuité** si on connaît une borne supérieure à la pénalité maximale qui puisse être appliquée dans le terme de voisinage entre deux sites d'étiquettes différentes.

Exemple 6.3.
Si nous avons $V_{\{p,q\}}(x_p, x_q) = \min\{K, \|x_p - x_q\|\}$, où K est une constante positive, alors l'énergie de l'équation 6.12 préserve la discontinuité.

Il est à noter que si $V_{\{p,q\}}$ est convexe, alors l'énergie correspondante ne préserve pas la discontinuité. Avoir deux étiquettes "très" différentes attribuées à deux sites voisins entraînerait une forte hausse de l'énergie. Pour avoir une énergie plus faible, il devra alors y avoir une ou plusieurs étiquettes "intermédiaires" pour les séparer.

Pour plus de détails, voir les livres [138, 210].

6.1.3.2 Modèle de Potts

Le **modèle de Potts** (du nom de son créateur, Renfrey Burnard Potts, voir [170]) est un cas particulier de l'énergie présentée dans l'équation 6.12 où le terme de voisinage est donné par :

$$V_{\{p,q\}}(x_p, x_q) = \lambda \mathbb{1}(x_p \neq x_q) \tag{6.13}$$

où

$$\mathbb{1}(k) = \begin{cases} 1 & \text{si la condition } k \text{ est vraie ;} \\ 0 & \text{sinon.} \end{cases} \tag{6.14}$$

Ce modèle, introduit en vision par ordinateur en 1990 dans [89], pénalise donc de manière égale la discontinuité d'étiquette entre toute paire de sites voisins. Ceci a pour conséquence de réduire la longueur des frontières entre les régions (i.e. les différents ensembles de sites $\mathcal{P}_\ell$ pour $\ell \in \mathcal{L}$) par rapport à ce qu'aurait pu donner le terme direct uniquement, ce qui correspond donc à un **lissage spatial**. Plus la constante λ est grande par rapport aux termes directs, plus le lissage aura d'influence sur la détermination de l'étiquetage.

Le **modèle de Potts généralisé** correspond au modèle de Potts dans lequel on remplace la constante λ par une variable dépendant de la paire de sites. Ainsi on obtient le terme de voisinage suivant :

$$V_{\{p,q\}}(x_p, x_q) = \lambda_{\{p,q\}} \mathbb{1}(x_p \neq x_q) \tag{6.15}$$

Ce modèle ne représente plus un lissage spatial mais ne dépend toujours pas des étiquettes attribuées aux deux sites considérés.

Il est à noter que ces modèles correspondent à des énergies préservant la discontinuité.

6.2 Résolution par coupes de graphes

Les méthodes de minimisation d'énergie par coupes de graphes consistent à traduire le problème sous forme d'un ou plusieurs graphes successifs sur lesquels sont calculées des coupes minimales dont on pourra déduire une réalisation du problème d'étiquetage considéré offrant une énergie minimale ou, à défaut, une bonne approximation de celle-ci.

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 dans [103] dans la cadre d'une restoration d'image binaire en utilisant le modèle de Potts (voir sous-section 6.2.1). Leur objectif était de mettre en évidence que la méthode de minimisation par recuit simulé (voir section 6.3.1) 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 [35, 36, 37, 38].

En 2002, Vladimir Kolmogorov et Ramin Zabih énoncèrent dans [123] 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 comme nous le verrons plus loin.

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 [87]. Néanmoins, dans le cadre de ce mémoire, nous nous en tiendrons au cas de cliques à deux sites uniquement (autrement dit les fonctions de type $\mathcal{F}^2$ telle que définies dans [87, 123]).

Nous avons vu dans le chapitre 5 que les poids d'un graphe sur lequel calculer une coupe minimale sont nécessairement positifs. Cependant, outre le fait que certains algorithmes de calcul de flot maximal s'accomodent en fait très bien de poids négatifs (comme, par exemple, celui proposé par Yuri Boykov et Vladimir Kolmogorov dans [32, 33]), il est à préciser que la remarque 6.2 nous permet, à partir d'une énergie ayant des termes négatifs, de nous ramener au cas où tous les termes sont positifs en rajoutant des constantes suffisamment grandes aux termes concernés pour que leurs valeurs soient positifs.

Il est à noter que, pour certaines méthodes de résolution par coupes minimales, des arcs ou des arêtes de poids infinis (notés $+\infty$) sont considérés. Cependant, comme dans le chapitre 5, cela représente simplement un poids très supérieur à la somme des poids finis du graphe considéré. Ainsi, une coupe minimale ne pourra jamais contenir un arc

de poids infini.

Tout algorithme de calcul de coupe minimale est susceptible de convenir à ces méthodes de résolution, mais, comme déjà expliqué dans la sous-section 5.3.1.4, l'algorithme proposé par Yuri Boykov et Vladimir Kolmogorov dans [32, 33] est particulièrement adapté à ce type de problème.

Un tour d'horizon des méthodes de minimisation par coupes de graphes présentées ici peut également être trouvé dans [34].

6.2.1 Cas particulier de l'étiquetage binaire

Le cas de l'étiquetage binaire (par exemple $\mathcal{L} = \{0,1\}$) associé à la minimisation d'une énergie du type de l'équation 6.12 correspond au **modèle de Ising** (du nom de son créateur, le physicien Ernst Ising) et est un cas particulier du modèle de Potts.
La méthode de minimisation présentée ci-après garantit l'obtention d'un minimum global.

Soit un problème d'étiquetage binaire de $\mathcal{L} = \{0,1\}$ sur $\mathcal{P}$ lié à l'énergie E de l'équation 6.12 tel que, pour tout site $p \in \mathcal{P}$, nous avons $\mathcal{L}_p = \{0,1\}$ (autrement dit, chaque site est susceptible d'être étiqueté 0 ou 1).
Le graphe orienté $G = (S, A)$ pondéré sur les arcs correspondant à ce problème est construit de la façon suivante :

- $S = \mathcal{P} \cup \{0,1\}$:
  - un sommet pour chaque site de $\mathcal{P}$, et
  - un sommet source 0 et un sommet puits 1 ;
- $A = \left\{ \cup_{\{p,q\}\in\mathcal{V}} \left\{(p,q) \cup (q,p)\right\}\right\} \cup \left\{ \cup_{p\in\mathcal{P}} \left\{(0,p) \cup (p,1)\right\}\right\}$:
  - des arcs symétriques, appelées v-arcs, reliant chaque paire de sites $p, q \in \mathcal{P}$ qui sont voisins ($\{p,q\} \in \mathcal{V}$) :
    - le v-arc (p,q), pondéré par $V_{\{p,q\}}(0,1)$, et
    - le v-arc (q,p), pondéré par $V_{\{p,q\}}(1,0)$;
  - des arcs, appelées t-arcs, reliant :
    - la source 0 à chaque sommet $p \in \mathcal{P}$, pondéré par $D_p(1)$,
    - chaque sommet $p \in \mathcal{P}$ au puits 1, pondéré par $D_p(0)$.

Une fois une coupe minimale C calculée dans le graphe $G = (S, A)$, chaque site $p \in \mathcal{P}$ du graphe $G' = (S, A \setminus C)$ n'est plus connecté qu'à la source 0 ou au puits 1 puisque la coupe contient un seul de ces deux arcs. En conséquence de quoi on déduit le résultat du problème d'étiquetage comme suit :

- si le site p est connecté à la source 0, alors p est étiqueté 0 ($x_p = 0$) ;
- si le site p est connecté au puits 1, alors p est étiqueté 1 ($x_p = 1$).

Une illustration des différentes coupes possibles sur un sous-graphe induit par la source 0, le puits 1 et deux sites $p, q \in \mathcal{P}$ voisins (i.e. $\{p,q\} \in \mathcal{V}$) se trouve dans la figure 6.2.

Une illustration de problème d'étiquetage binaire avec résolution par coupe minimale se trouve dans la figure 6.3.

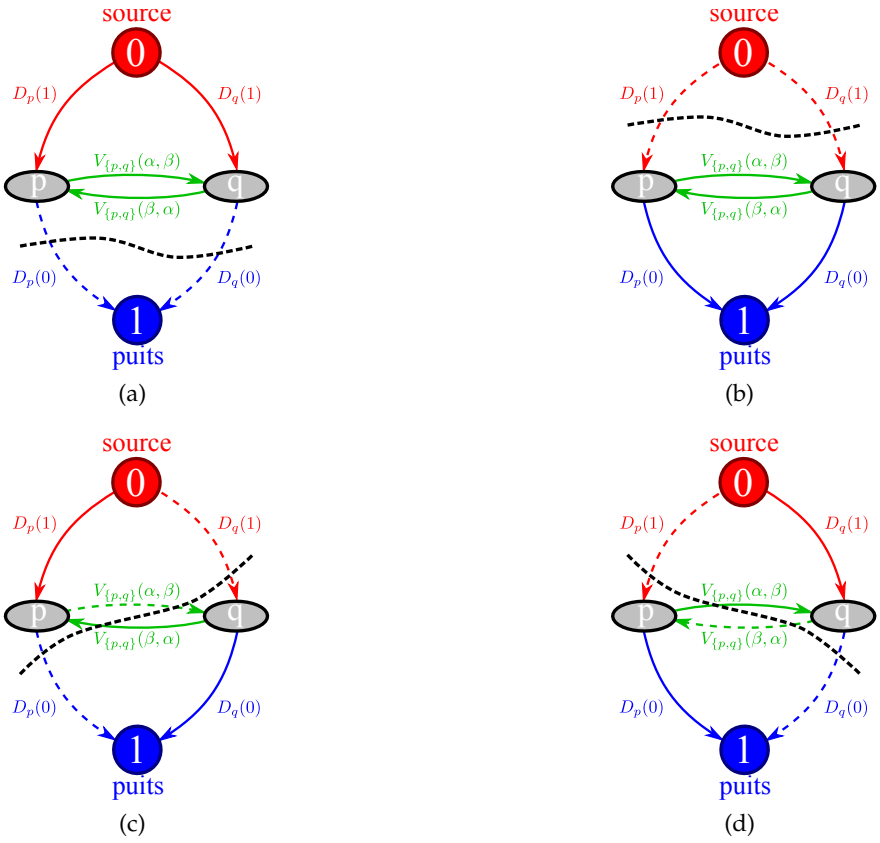


FIGURE 6.2 – Configurations possibles de coupes du sous-graphe induit par la source 0, le puits 1 et deux sites voisins $p, q \in \mathcal{P}$ dans un problème d'étiquetage binaire associé à la minimisation d'une énergie du type de l'équation 6.12.

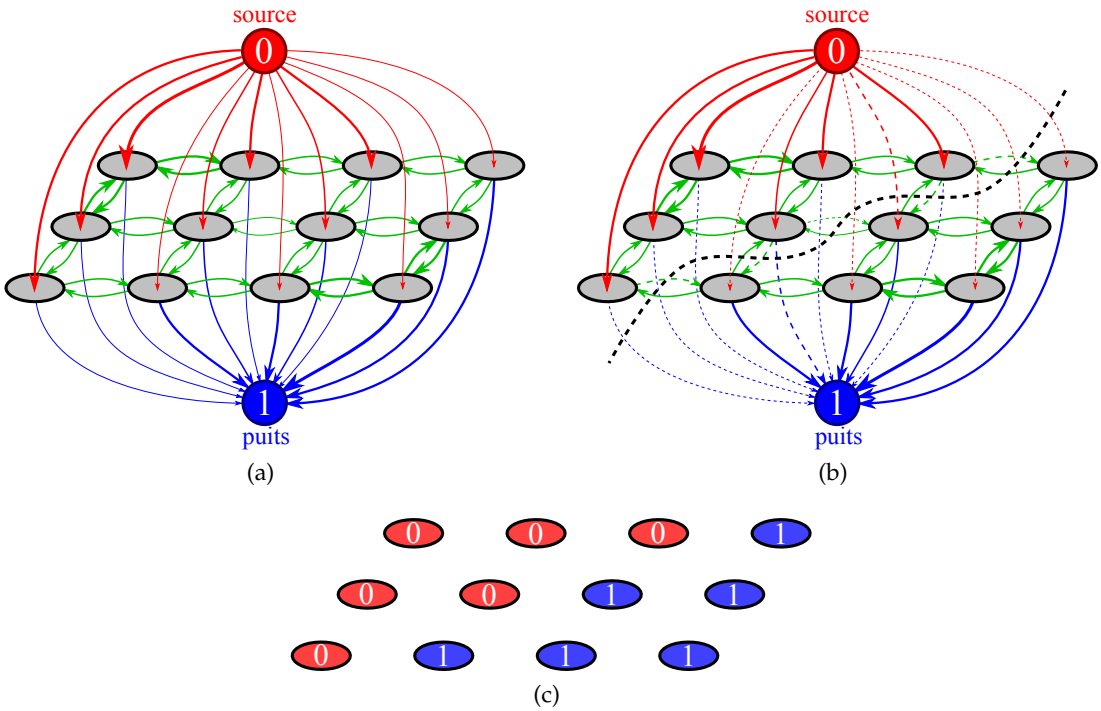


FIGURE 6.3 – (a) Graphe orienté G représentant l'énergie associée à un problème d'étiquetage binaire (l'épaisseur des arcs représente le poids appliqué). (b) Coupe minimale C sur le graphe G (la ligne pointillée traversant les arcs pointillés). (c) Résultat du problème d'étiquetage correspondant à la coupe C.

En 2002, Vladimir Kolmogorov et Ramin Zabih mirent en évidence dans [123] une condition nécessaire et suffisante à la résolution par coupe minimale de graphe d'un problème d'étiquetage binaire.

L'énergie E d'un problème d'étiquetage binaire est **représentable par graphe** s'il existe un graphe $G = (S, A)$ avec une source 0, un puits 1 et $\mathcal{P} = \{p_1, \ldots, p_n\} \subseteq S \setminus \{0, 1\}$ tel que, pour toute configuration $x = \{x_1, \ldots, x_n\} \in \mathcal{X}$, il existe une coupe C avec :
- $\forall i \in [1; n]$, si $x_i = 0$, alors p_i est dans la composante connexe contenant la source 0 ;
- $\forall i \in [1; n]$, si $x_i = 1$, alors p_i est dans la composante connexe contenant le puits 1 ;
- $E(x) = P(C) + k$ où k représente une constante.

Si $k = 0$, alors on dit que l'énergie E est **représentée exactement par le graphe** G.

Le théorème 6.4 définit l'ensemble des énergies qui sont représentables par graphe.

Théorème 6.4 ([123], théorème 4.1).
*Le problème d'étiquetage binaire avec $\mathcal{L} = \{0, 1\}$ associé à une énergie E du type de l'équation 6.12 est représentable par graphe si et seulement si la fonction $V_{\{p,q\}}$, représentant le terme de voisinage, respecte la **condition de régularité** suivante :*

$$V_{\{p,q\}}(0,0) + V_{\{p,q\}}(1,1) \leq V_{\{p,q\}}(0,1) + V_{\{p,q\}}(1,0) \tag{6.16}$$

On appelle fonction **régulière** (pour $\mathcal{L}$), ou fonction **sous-modulaire** (pour $\mathcal{L}$), une fonction satisfaisant la condition de régularité sur $\mathcal{L}$.

Remarque 6.5.
Il est à noter que la condition de régularité (voir théorème 6.4) met en évidence le fait que des énergies anisotropes (i.e. dont le terme de voisinage dépend de l'orientation) peuvent être minimisées sur des graphes par coupe minimale.

Une construction de graphe optimisée et valable pour toute énergie ayant un terme de voisinage régulier fut présentée dans ce même article [123]. Ce graphe a la particularité d'avoir moins d'arcs que dans le cas de la construction "basique" décrite précédemment et donc de permettre une recherche de coupe minimale plus rapide. Sa construction repose sur une optimisation séparée des termes directs et des termes de voisinage. Ainsi, on construit séparément les graphes pour les termes directs et pour les termes de voisinage, que l'on "fusionne" ensuite en prenant l'union des arcs des graphes et en additionnant les poids des arcs en commun. Cette fusion est possible grâce au théorème 6.6.

Théorème 6.6 ([123], appendix B)**.**
La somme de deux énergies représentables par graphes est représentable par graphe.

La construction des graphes pour chacun des termes d'une énergie à terme de voisinage régulier du type de l'équation 6.12 se fait comme suit :

- **Terme direct**
  Pour tout site $p \in \mathcal{P}$:
  - si $D_p(1) > D_p(0)$ créer un arc $(0, p)$ de poids $D_p(1) - D_p(0)$,
  - sinon créer un arc $(p, 1)$ de poids $D_p(0) - D_p(1)$.
- **Terme de voisinage :**
  Pour toute paire de sites voisins $p, q \in \mathcal{P}$ (i.e. $\{p, q\} \in \mathcal{V}$) :
  - créer un arc (p, q) de poids $V_{\{p,q\}}(0,1) + V_{\{p,q\}}(1,0) - V_{\{p,q\}}(0,0) - V_{\{p,q\}}(1,1)$;
  - pour le site p :
    - si $V_{\{p,q\}}(1,0) > V_{\{p,q\}}(0,0)$ créer un arc $(0, p)$ de poids $V_{\{p,q\}}(1,0) - V_{\{p,q\}}(0,0)$,
    - sinon créer un arc $(p, 1)$ de poids $V_{\{p,q\}}(0,0) - V_{\{p,q\}}(1,0)$;
  - pour le site q :
    - si $V_{\{p,q\}}(1,1) > V_{\{p,q\}}(1,0)$ créer un arc $(0, p)$ de poids $V_{\{p,q\}}(1,1) - V_{\{p,q\}}(1,0)$,
    - sinon créer un arc $(p, 1)$ de poids $V_{\{p,q\}}(1,0) - V_{\{p,q\}}(1,1)$.

Remarque 6.7.
Si l'on considère que les termes de l'énergie représentée sont tous positifs (ou ramenés au cas positif comme expliqué en début de section 6.2), alors, par construction, tous les poids des arcs créés sont positifs (pour le poids de l'arc (p, q), cela est garanti par la condition de régularité du théorème 6.4).

Des illustrations des différentes configurations de graphes possibles se trouvent dans la figure 6.4 pour le terme direct et dans la figure 6.6 pour le terme de voisinage.

Il est à noter que ces constructions trouvent leur justification dans les remarques 5.3, 5.9 et 6.2. En effet, comme le montrent respectivement les figures 6.5 et 6.7, dans chacun des cas exposés, toutes les coupes ont été additionnées à une constante bien choisie qui permet ainsi d'obtenir des poids d'arcs nuls et qui peuvent donc être supprimés du graphe par rapport à la construction "classique" vue précédemment.

6.2.2 Cas général

Bien qu'offrant un minimum global de l'énergie dans le cadre d'un étiquetage binaire, la minimisation d'une énergie du type de l'équation 6.12 par coupe minimale sur un graphe resta ignorée pendant près de 10 ans avant d'être généralisée à des problèmes ayant plus de deux étiquettes permettant d'obtenir, à défaut du minimum global, une bonne approximation de celui-ci.

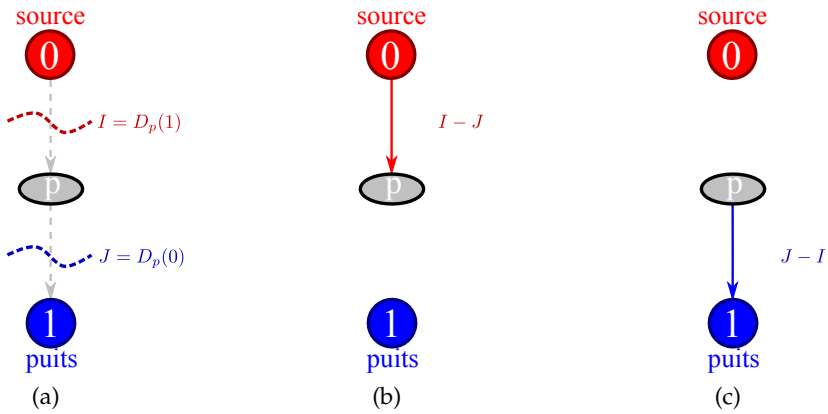


FIGURE 6.4 – (a) Différentes coupes possibles dans un sous-graphe de résolution lié au terme direct dans un problème d'étiquetage binaire associé à la minimisation d'une énergie du type de l'équation 6.12 (introduction d'une nouvelle notation pour le poids des coupes). (b,c) Configurations possibles du sous-graphe optimisé dans un problème d'étiquetage binaire associé à la minimisation d'une énergie du type de l'équation 6.12 avec : (b) $I - J \geq 0$, (c) $J - I \geq 0$.

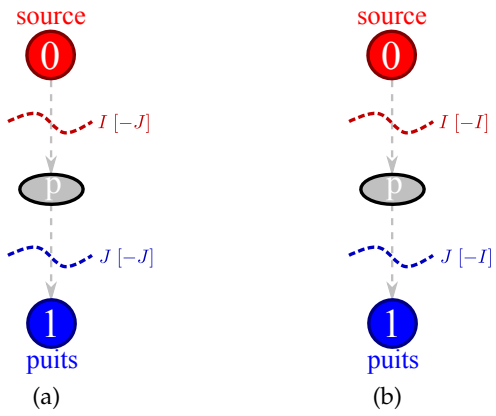


FIGURE 6.5 – Ajouts de constantes aux différentes coupes possibles pour optimiser le sous-graphe de résolution lié au terme direct dans un problème d'étiquetage binaire associé à la minimisation d'une énergie du type de l'équation 6.12 pour, respectivement, les figures 6.4b et 6.4c.

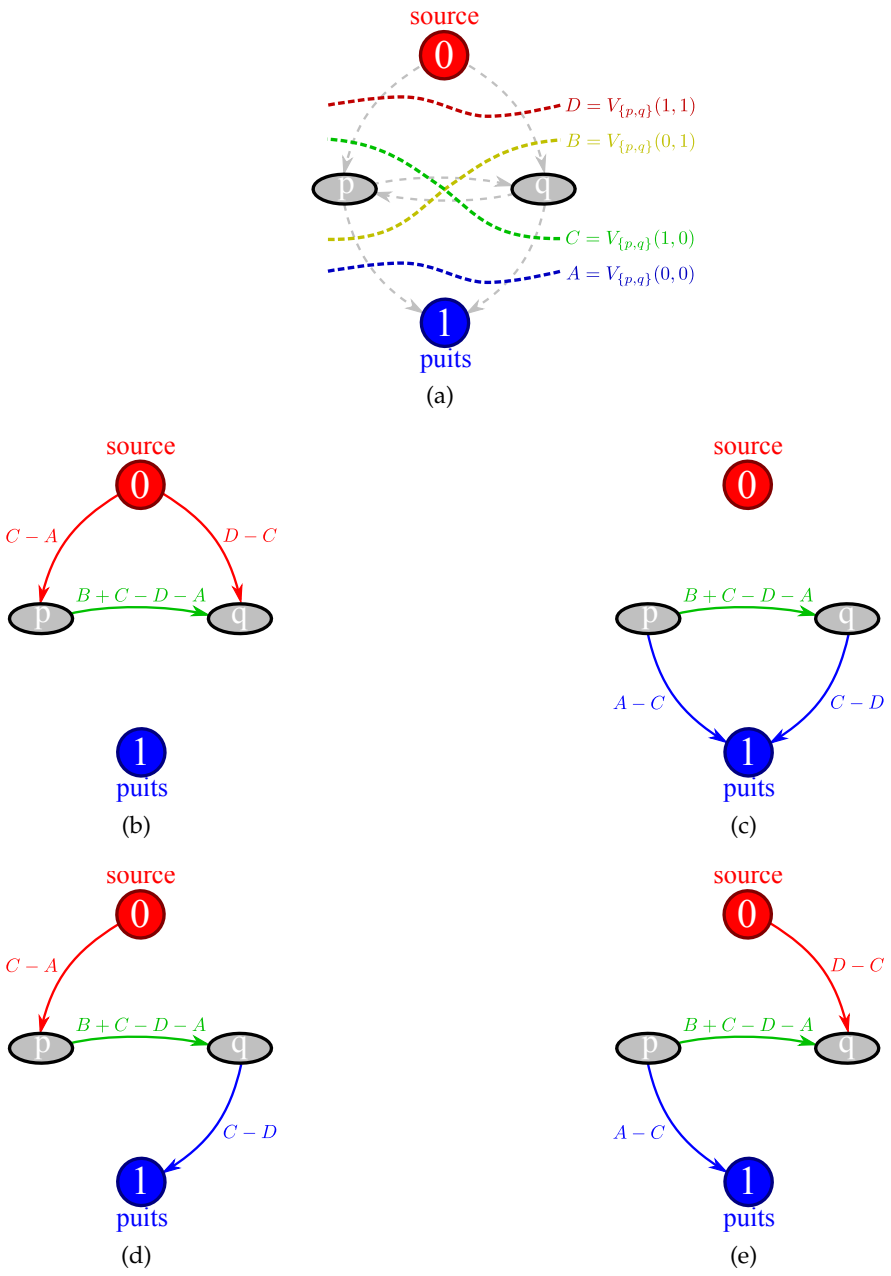


FIGURE 6.6 – (a) Différentes coupes possibles dans un sous-graphe de résolution lié au terme de voisinage dans un problème d'étiquetage binaire associé à la minimisation d'une énergie du type de l'équation 6.12 (introduction d'une nouvelle notation pour le poids des coupes). (b) à (e) Configurations possibles du sous-graphe optimisé dans un problème d'étiquetage binaire associé à la minimisation d'une énergie du type de l'équation 6.12 avec : (b) $C - A \geq 0$ et $D - C \geq 0$, (c) $A - C \geq 0$ et $C - D \geq 0$, (d) $C - A \geq 0$ et $C - D \geq 0$, (e) $A - C \geq 0$ et $D - C \geq 0$.

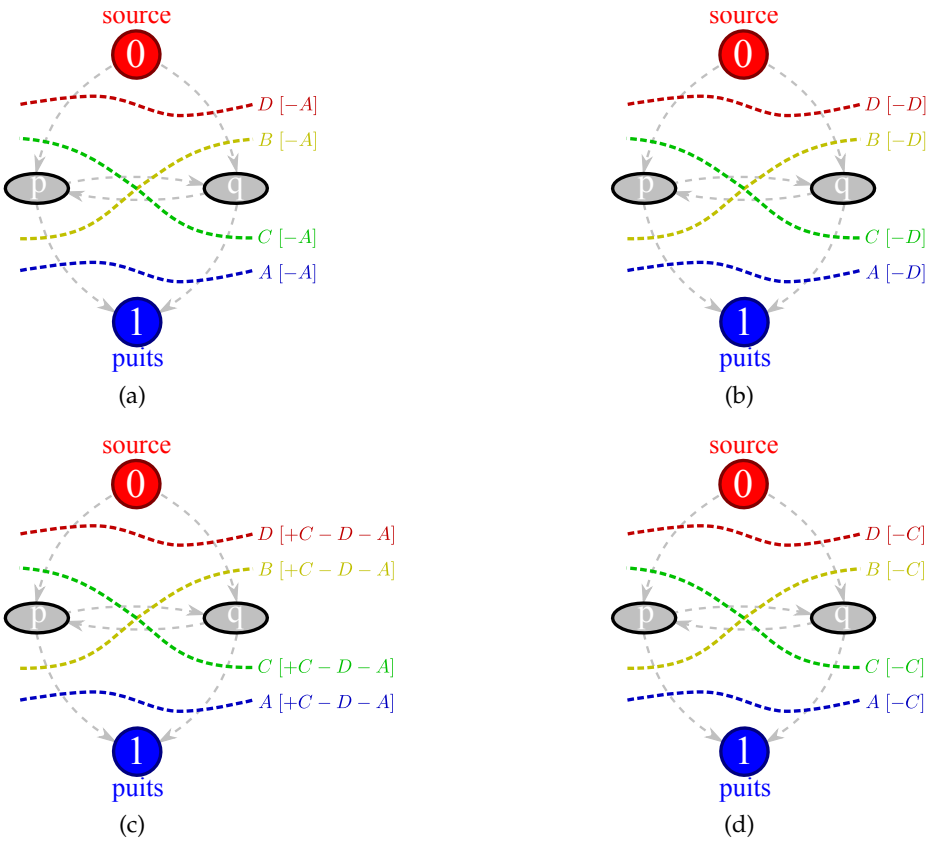


FIGURE 6.7 – Ajouts de constantes aux différentes coupes possibles pour optimiser le sous-graphe de résolution lié au terme de voisinage dans un problème d'étiquetage binaire associé à la minimisation d'une énergie du type de l'équation 6.12 pour, respectivement, les figures 6.6b, 6.6c, 6.6d et 6.6e.

Un problème d'étiquetage de $\mathcal{L}$ sur $\mathcal{P}$ lié à l'énergie E de l'équation 6.12 peut se représenter sous la forme d'un graphe non-orienté $G = (S, A)$ avec :

- $S = \mathcal{P} \cup \mathcal{L}$:
  - un sommet pour chaque site de $\mathcal{P}$, et
  - un sommet, appelé terminal, pour chaque étiquette de $\mathcal{L}$;
- $A = \mathcal{V} \cup \left\{ \cup_{p \in \mathcal{P}} \cup_{\ell_i \in \mathcal{L}_p} \{p, \ell_i\} \right\}$:
  - des arêtes, appelées v-arêtes, reliant chaque paire de sites voisins de $\mathcal{V}$, et
  - des arêtes, appelées t-arêtes, reliant chaque sommet $p \in \mathcal{P}$ à tous les terminaux de $\mathcal{L}_p$.

Il est à noter que, par définition d'un problème d'étiquetage, tout site $p \in \mathcal{P}$ est relié à au moins un terminal par une t-arête puisqu'il doit pouvoir être étiqueté par au moins une étiquette ($\forall p \in \mathcal{P}, \mathcal{L}_p \neq \emptyset$).

Une illustration d'une représentation par graphe d'un problème d'étiquetage se trouve dans la figure 6.8.

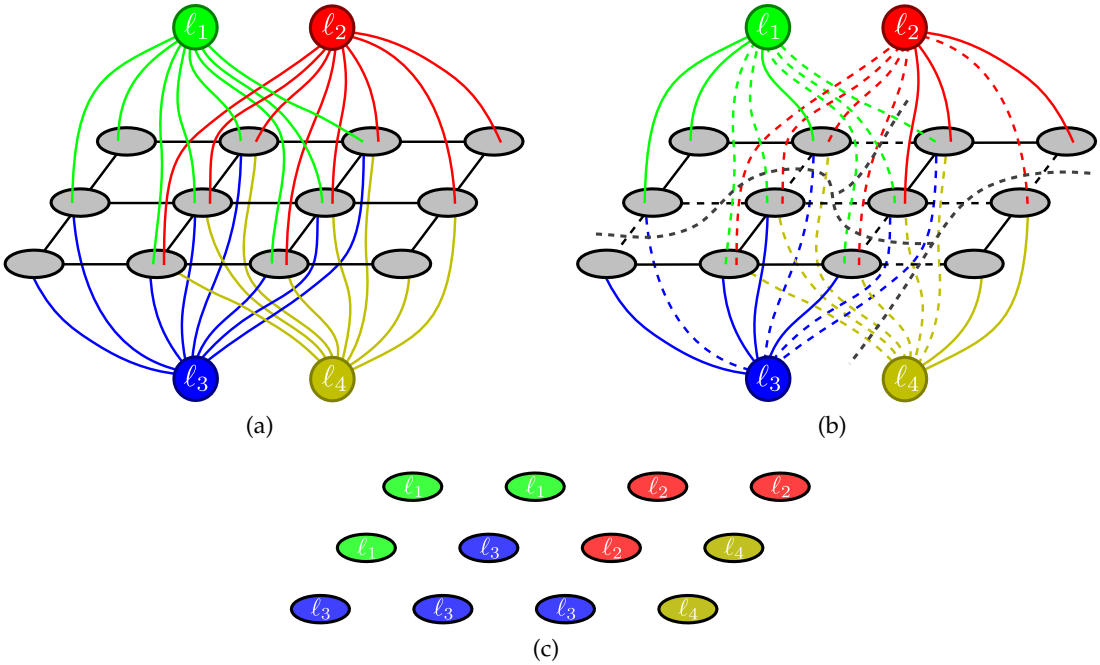


FIGURE 6.8 – (a) Graphe non-orienté G représentant un problème d'étiquetage. (b) *Multiway cut* C (la ligne pointillée traversant les arêtes pointillées) sur le graphe G. (c) Résultat du problème d'étiquetage correspondant à la *multiway cut* C.

Dans un graphe $G = (S, A)$ associé à un problème d'étiquetage, on appelle *multiway cut* **réalisable** une *multiway cut* du graphe G telle que chaque sommet de $G' = (S, A \setminus C)$ représentant un site n'appartient qu'à une seule t-arête qui le relie à un unique terminal.

On remarque qu'une *multiway cut* réalisable C d'un tel graphe permet de définir un résultat d'étiquetage en attribuant l'étiquette d'un terminal à tous les sites auxquels ce dernier est relié dans $G' = (S, A \setminus C)$.

Il est à noter que, sauf cas particulier (voir sous-section 6.2.3.2), aucune application de poids n'est faite directement sur ce graphe puisque le terme de voisinage dépend, dans

la plupart des cas, de l'étiquetage des sites.

En 1998 et en 1999, Yuri Boykov, Olga Veksler et Ramin Zabih développèrent deux algorithmes permettant de trouver un minimum local qui est une bonne approximation du minimum global d'un problème d'étiquetage à n étiquettes ($n > 2$) en se ramenant à des itérations du cas binaire décrit dans la sous-section 6.2.1. Les deux méthodes diffèrent sur la construction de graphe permettant de se ramener au cas binaire. Dans la première, celui-ci est utilisé sous le nom d'$\{\alpha, \beta\}$-*swap* et, dans la seconde, sous le nom d'α-*expansion*. Ces deux constructions furent décrites initialement dans, respectivement, [35] et [37] (il est à noter que, dans ces articles, ils ne portaient pas encore ces noms). Ces méthodess n'étaient initialement destinées à minimiser que des énergies correspondant au modèle de Potts généralisé. Ils firent l'objet d'une comparaison et d'une généralisation à davantage d'énergies du type de l'équation 6.12 dans [36, 38].

Une itération d'$\{\alpha, \beta\}$-*swap* ou d'α-*expansion* consiste en une modification de l'étiquetage d'entrée $x \in \mathcal{X}$ en un étiquetage $x' \in \mathcal{X}$ de sorte à en diminuer l'énergie ($E(x') \leq E(x)$). La première permet de passer à l'étiquette β des sites auparavant étiquetés α et inversement alors que la seconde permet de passer à l'étiquette α des sites ayant auparavant une étiquette autre que α.

La résolution d'un problème d'étiquetage par minimisation d'énergie avec l'une ou l'autre de ces méthodes se déroule en faisant décroître l'énergie par itérations successives jusqu'à stabilité. L'énergie correspondant à l'étiquetage qui en résulte est un minimum local proche du minimum global.

Les sous-sections 6.2.2.1 et 6.2.2.2 donneront plus de détails sur les conditions et les exécutions de l'$\{\alpha, \beta\}$-*swap* et, respectivement, de l'α-*expansion*.

On note $\mathcal{P}_{\alpha(\beta)} = \{p \in \mathcal{P}_\alpha \mid \beta \in \mathcal{L}_p\}$ l'ensemble des sites étiquetés α et susceptibles d'être étiquetés β.

6.2.2.1 $\{\alpha, \beta\}$-*swap*

Soit $\mathcal{P}_{\alpha\beta} = \mathcal{P}_{\alpha(\beta)} \cup \mathcal{P}_{\beta(\alpha)}$ l'ensemble des sites étiquetés α et susceptibles d'être étiquetés β ou inversement. Par définition, nous savons donc que, dans le graphe G correspondant, chaque site $p \in \mathcal{P}_{\alpha\beta}$ est relié au terminal α et au terminal β.

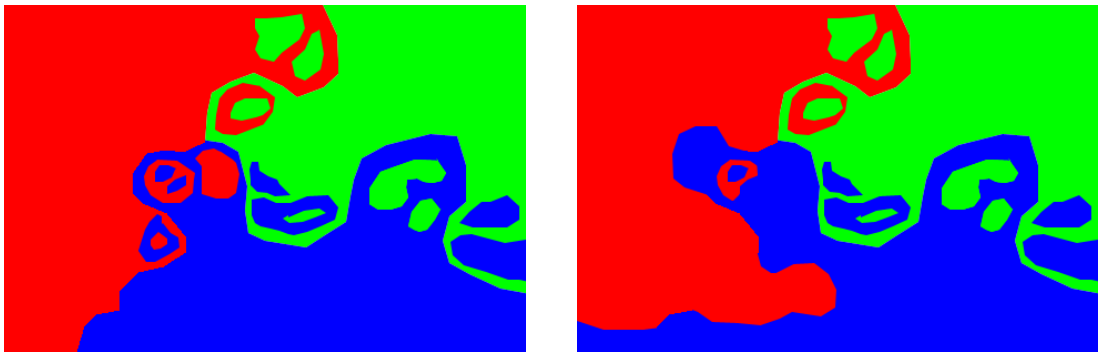
En partant d'un étiquetage quelconque $x \in \mathcal{X}$, un $\{\alpha, \beta\}$-*swap* consiste, pour deux étiquettes $\alpha, \beta \in \mathcal{L}$, à modifier l'étiquetage des sites $p \in \{\mathcal{P}_\alpha \cup \mathcal{P}_\beta\}$ en étiquetant α des sites de $\mathcal{P}_\beta$ et inversement.

Une illustration de modification d'étiquetage par $\{\alpha, \beta\}$-*swap* se trouve dans la figure 6.9.

Notre objectif étant de minimiser une énergie, nous chercherons donc à obtenir, parmi les configurations qu'il est possible d'atteindre par cet échange d'étiquettes, celle qui minimise le plus l'énergie.

Le graphe orienté $G_{\alpha\beta} = (S_{\alpha\beta}, A_{\alpha\beta})$ pondéré sur les arcs correspondant à un $\{\alpha, \beta\}$-*swap* est construit de la façon suivante :

- $S_{\alpha\beta} = \mathcal{P}_{\alpha\beta} \cup \{\alpha, \beta\}$:
  - un sommet pour chaque site de $\mathcal{P}_{\alpha\beta}$,
  - un sommet source α et un sommet puits β ;



(a) (b)

FIGURE 6.9 – (a) Etiquetage des pixels d'une image avec trois étiquettes (*rouge*, *verte* et *bleue*). (b) Etiquetage des pixels après une opération de *(rouge,bleue)-swap* (la région étiquetée *verte* n'a pas été modifiée).

- $A_{\alpha\beta} = \left\{ \cup_{\substack{p,q\in\mathcal{P}_{\alpha\beta} \\ \{p,q\}\in\mathcal{V}}} \{(p,q)\cup(q,p)\} \right\} \cup \left\{ \cup_{p\in\mathcal{P}_{\alpha\beta}} \{(\alpha,p)\cup(p,\beta)\} \right\}$:
  - des arcs symétriques, appelées v-arcs, reliant chaque paire de sites $p,q \in \mathcal{P}_{\alpha\beta}$ qui sont voisins (i.e. $\{p,q\} \in \mathcal{V}$) :
    - le v-arc (p,q), pondéré par $V_{\{p,q\}}(\alpha,\beta)$, et
    - le v-arc (q,p), pondéré par $V_{\{p,q\}}(\beta,\alpha)$;
  - des arcs, appelées t-arcs, reliant :
    - la source α à chaque sommet $p \in \mathcal{P}_{\alpha\beta}$, pondéré par $D_p(\beta) + \sum_{\substack{q\in\mathcal{V}_p \\ q\notin\mathcal{P}_{\alpha\beta}}} V_{\{p,q\}}(\beta,x_q)$,
    - chaque sommet $p \in \mathcal{P}_{\alpha\beta}$ au puits β, pondéré par $D_p(\alpha) + \sum_{\substack{q\in\mathcal{V}_p \\ q\notin\mathcal{P}_{\alpha\beta}}} V_{\{p,q\}}(\alpha,x_q)$.

La construction du graphe $G_{\alpha\beta}$ est donc très similaire à celle pour le cas binaire, à l'exception des sites $p \in \mathcal{P}_{\alpha\beta}$ qui ont un voisin $q \notin \mathcal{P}_{\alpha\beta}$ et pour lesquels les termes de voisinage possibles pour les étiquettes α et β doivent être inclus dans les t-arcs.

Remarque 6.8.
Il est à noter que la fermeture symétrique du graphe $G_{\alpha\beta}$ correspond exactement au sous-graphe de G induit par $\mathcal{P}_{\alpha\beta} \cup \alpha \cup \beta$.

Une fois une coupe minimale C calculée dans le graphe $G_{\alpha\beta} = (S_{\alpha\beta}, A_{\alpha\beta})$, chaque site $p \in \mathcal{P}_{\alpha\beta}$ du graphe $G'_{\alpha\beta} = (S_{\alpha\beta}, A_{\alpha\beta} \setminus C)$ n'est plus connecté qu'à la source α ou au puits β puisque la coupe contient un seul de ces deux arcs. En conséquence de quoi on déduit le résultat du problème d'étiquetage comme suit :

- si le site p est connecté à la source α, alors p est étiqueté α (i.e. $x_p = \alpha$) ;
- si le site p est connecté au puits β, alors p est étiqueté β (i.e. $x_p = \beta$).

Remarque 6.9.
Il est à noter que la construction de graphe présentée ici diffère sensiblement des cas présentés dans [36, 35, 38]. En effet, dans ces articles, la construction est présentée sous forme d'un graphe non-orienté. De plus, les poids des t-arcs de la source et du puits ont été inversés ici afin de faciliter la lecture du résultat : dans notre cas il suffit d'attribuer l'étiquette correspondant à la source ou, respectivement, au puits pour tous les sites qui sont dans la composante connexe à laquelle elle, ou il, appartient dans le graphe auquel on aura retiré les arcs d'une coupe minimale alors que dans sa version d'origine il faut tester l'appartenance de chaque t-arc à la coupe pour attribuer au site correspondant son étiquette. Enfin, le cas où un site ne peut pas recevoir certaines étiquettes n'y est pas traité.

Une illustration des différentes coupes possibles sur un sous-graphe induit par la source α, le puits β et deux sites $p, q \in \mathcal{P}_{\alpha\beta}$ voisins (i.e. $\{p,q\} \in \mathcal{V}$) se trouve dans la figure 6.10.

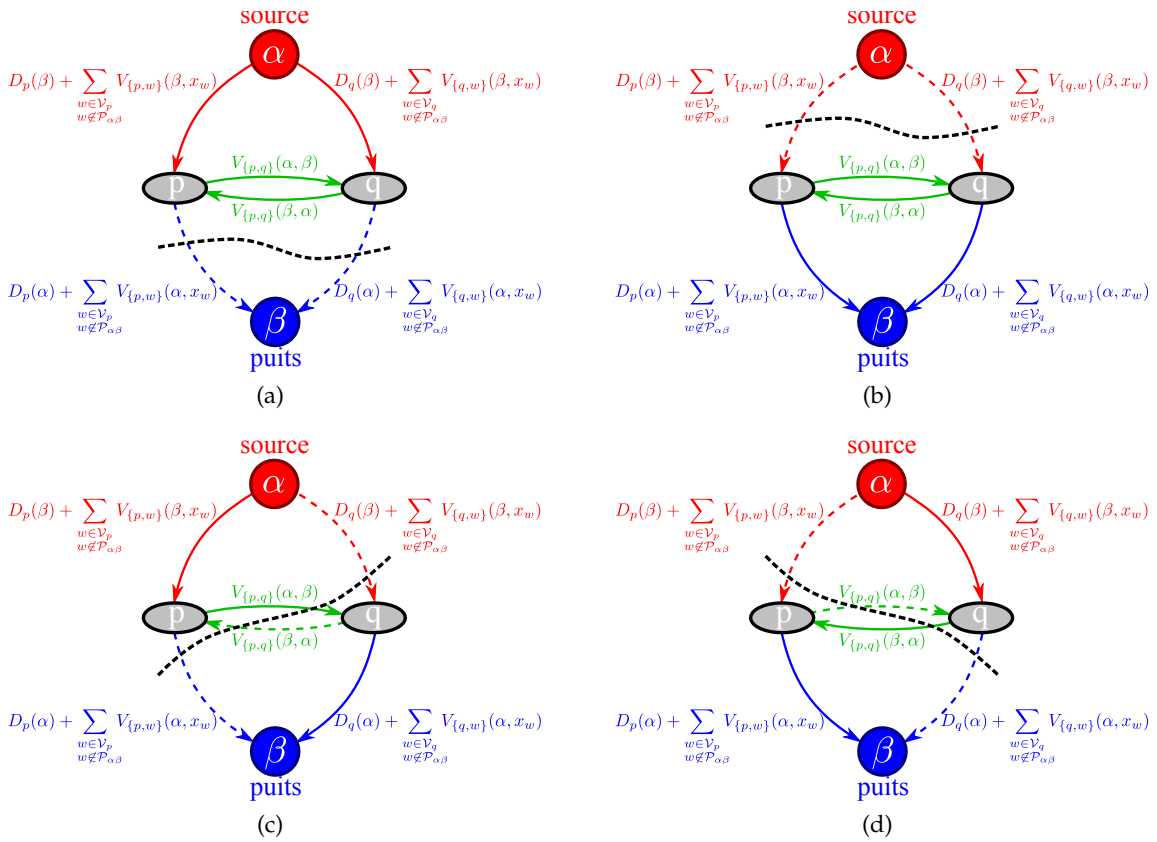


FIGURE 6.10 – Configurations possibles de coupes du sous-graphe induit par la source α, le puits β et deux sites voisins $p, q \in \mathcal{P}_{\alpha\beta}$ dans une itération d'$\{\alpha,\beta\}$-*swap* associée à la minimisation d'une énergie du type de l'équation 6.12.

Une illustration de problème d'étiquetage avec une itération d'$\{\alpha,\beta\}$-*swap* par coupe minimale se trouve dans la figure 6.11.

Chaque exécution d'$\{\alpha,\beta\}$-*swap* tente donc de modifier l'étiquetage d'entrée $x \in \mathcal{X}$ en un étiquetage $x' \in \mathcal{X}$ de sorte à en diminuer l'énergie correspondante ($E(x') \leq E(x)$).

Afin d'obtenir la meilleure approximation possible du minimum global d'un problème d'étiquetage par $\{\alpha,\beta\}$-*swaps*, il suffit de procéder comme suit :

1. Initialiser les sites avec un étiquetage quelconque $x \in \mathcal{X}$
2. Initialiser *succès* $= 0$
3. Pour toute paire d'étiquettes (distinctes) $\{\alpha,\beta\} \subset \mathcal{L}$:
   - 3-1 Chercher $\check{x} = \arg\min\{E(x')\}$ où $E(x')$ est une configuration obtenue par un $\{\alpha,\beta\}$-*swap* sur x
   - 3-2 Si $E(\check{x}) < E(x)$, alors remplacer $x = \check{x}$ et *succès* $= 1$
4. Si *succès* $= 1$, retourner à l'étape 2
5. Renvoyer x

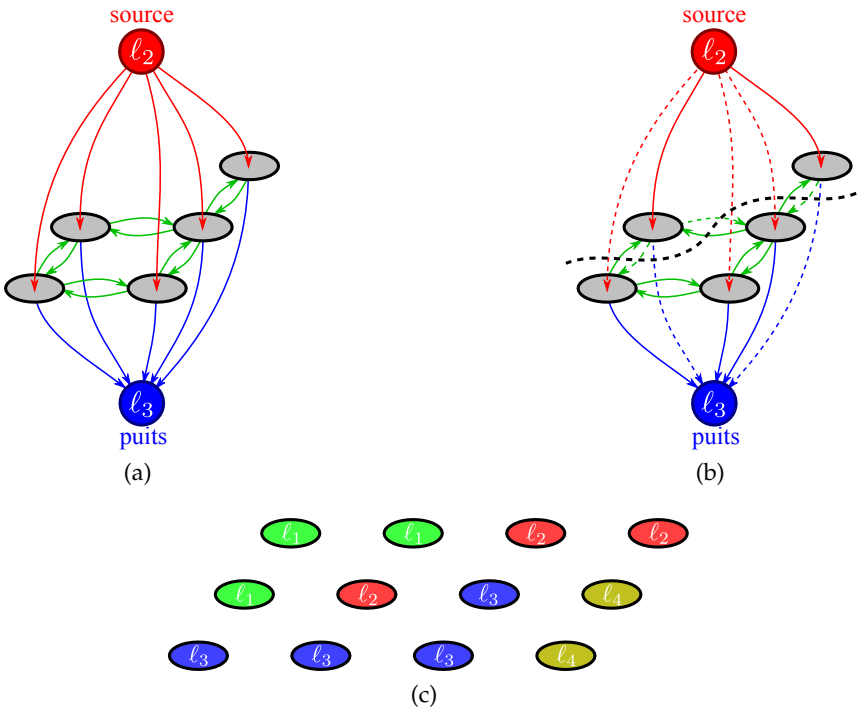


FIGURE 6.11 – (a) Graphe orienté $G_{rouge,bleu}$ pour le problème représenté dans la figure 6.8a et d'après l'étiquetage de la figure 6.8c. (b) Coupe minimale C sur le graphe $G_{rouge,bleu}$ (la ligne pointillée traversant les arcs pointillés). (c) Résultat du problème d'étiquetage correspondant à la coupe C.

On appelle **itération** une unique exécution des étapes 3.1-3.2 et **cycle** une exécution des étapes 2-4.

Un cycle comporte $\frac{n^2-n}{2}$ itérations (avec $n = |\mathcal{L}|$) puisque l'on considère une paire d'étiquettes distinctes et non un couple ordonné.

L'ordre d'exécution des itérations pour chaque paire d'étiquettes dans un cycle n'a pas d'importance. Un cycle est considéré comme "réussi" si au moins une itération a parmi de trouver un étiquetage strictement meilleur. L'algorithme s'arrête dès qu'un cycle n'a permis aucune amélioration et que l'étiquetage est stabilisé. Cet algorithme garantit de terminer en un nombre fini de cycles. En pratique, seuls quelques cycles sont nécessaires pour arriver à stabilité mais la majeure partie des changements d'étiquettes se produisent durant le premier cycle.

En 1999, il fut mis en évidence dans [36, 38] que, dans le cas général d'une énergie du type de l'équation 6.12, une condition suffisante pour permettre la minimisation par $\{\alpha,\beta\}$-*swaps* est que $V_{\{p,q\}}$ soit une semi-métrique sur $\mathcal{L}$. Nous en rappelons ci-dessous la définition.

La fonction $V_{\{p,q\}}$ est une **semi-métrique** sur l'espace des étiquettes $\mathcal{L}$ si $\forall \alpha,\beta \in \mathcal{L}$:

$$\begin{aligned} V_{\{p,q\}}(\alpha,\beta) = 0 \Leftrightarrow \alpha = \beta \\ V_{\{p,q\}}(\alpha,\beta) = V_{\{p,q\}}(\beta,\alpha) \geq 0 \end{aligned} \tag{6.17}$$

Cette condition permet donc d'utiliser les $\{\alpha,\beta\}$-*swaps* pour la minimisation de certaines énergies liées à un problème du type de celui de la sous-section 6.1.2.

Néanmoins, on déduit du théorème 6.4, mis en évidence en 2002 dans [123], une nouvelle condition, plus large que celle de semi-métrique, nécessaire et suffisante à la minimisation par $\{\alpha,\beta\}$-*swaps* d'une énergie du type de l'équation 6.12, présentée dans le corollaire 6.10.

Corollaire 6.10.
Un problème d'étiquetage peut être résolu pour l'énergie $E(x)$ (voir équation 6.12) à l'aide d'$\{\alpha,\beta\}$-swaps si et seulement si la fonction $V_{\{p,q\}}$, représentant le terme de voisinage, respecte la ***condition de régularité*** *suivante pour tout $\alpha,\beta \in \mathcal{L}_p \cap \mathcal{L}_q$:*

$$V_{\{p,q\}}(\alpha,\alpha) + V_{\{p,q\}}(\beta,\beta) \leq V_{\{p,q\}}(\alpha,\beta) + V_{\{p,q\}}(\beta,\alpha) \tag{6.18}$$

Il est à noter que si $V_{\{p,q\}}$ est une semi-métrique, alors $V_{\{p,q\}}$ respecte la condition de régularité du corollaire 6.10. Toutefois, cette condition est plus large que celle de semi-métrique, permettant, par exemple, la minimisation de problèmes d'étiquetage avec une énergie anisotrope. Cependant, il est à préciser que, dans ce cas, la résolution se fait exactement de la même façon que présentée précédemment.

La construction de graphe optimisée valable pour toute énergie ayant un terme de voisinage régulier présentée dans [123] est également applicable au cas de l'$\{\alpha,\beta\}$-*swap* (voir en fin de section 6.2.1). Sa construction se fait comme précédemment, excepté que l'on ne considère plus que les sites de $\mathcal{P}_{\alpha\beta}$ au lieu de tous les sites de $\mathcal{P}$.

La construction des graphes pour une itération d'$\{\alpha,\beta\}$-*swap*, avant fusion, pour chacun des termes d'une énergie à terme de voisinage régulier du type de l'équation 6.12 se fait de manière très similaire à précédemment. Nous ne donnerons donc pas davantage de détails et nous contenterons d'illustrer les différentes configurations de graphes possibles la figure 6.12 pour le terme direct et dans la figure 6.13 pour le terme de voisinage.

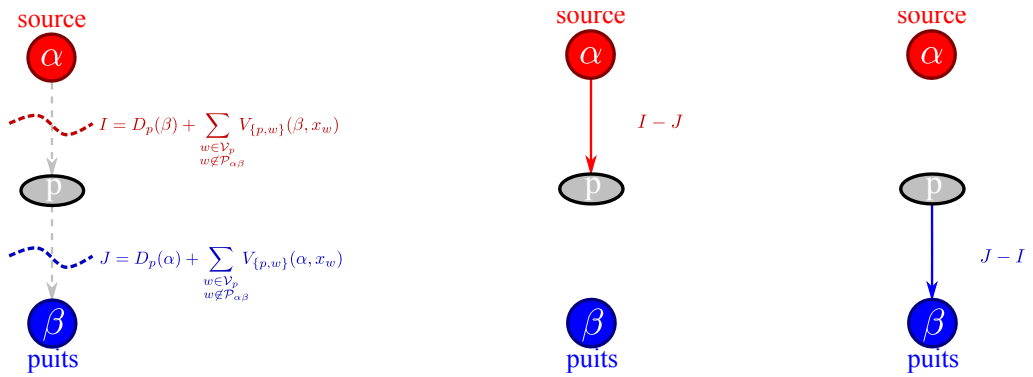


FIGURE 6.12 – (a) Différentes coupes possibles dans un sous-graphe de résolution lié au terme direct dans une itération d'$\{\alpha,\beta\}$-*swap* associée à la minimisation d'une énergie du type de l'équation 6.12 (introduction d'une nouvelle notation pour le poids des coupes). (b,c) Configurations possibles du sous-graphe optimisé dans une itération d'$\{\alpha,\beta\}$-*swap* associée à la minimisation d'une énergie du type de l'équation 6.12 avec : (b) $I - J \geq 0$, (c) $J - I \geq 0$.

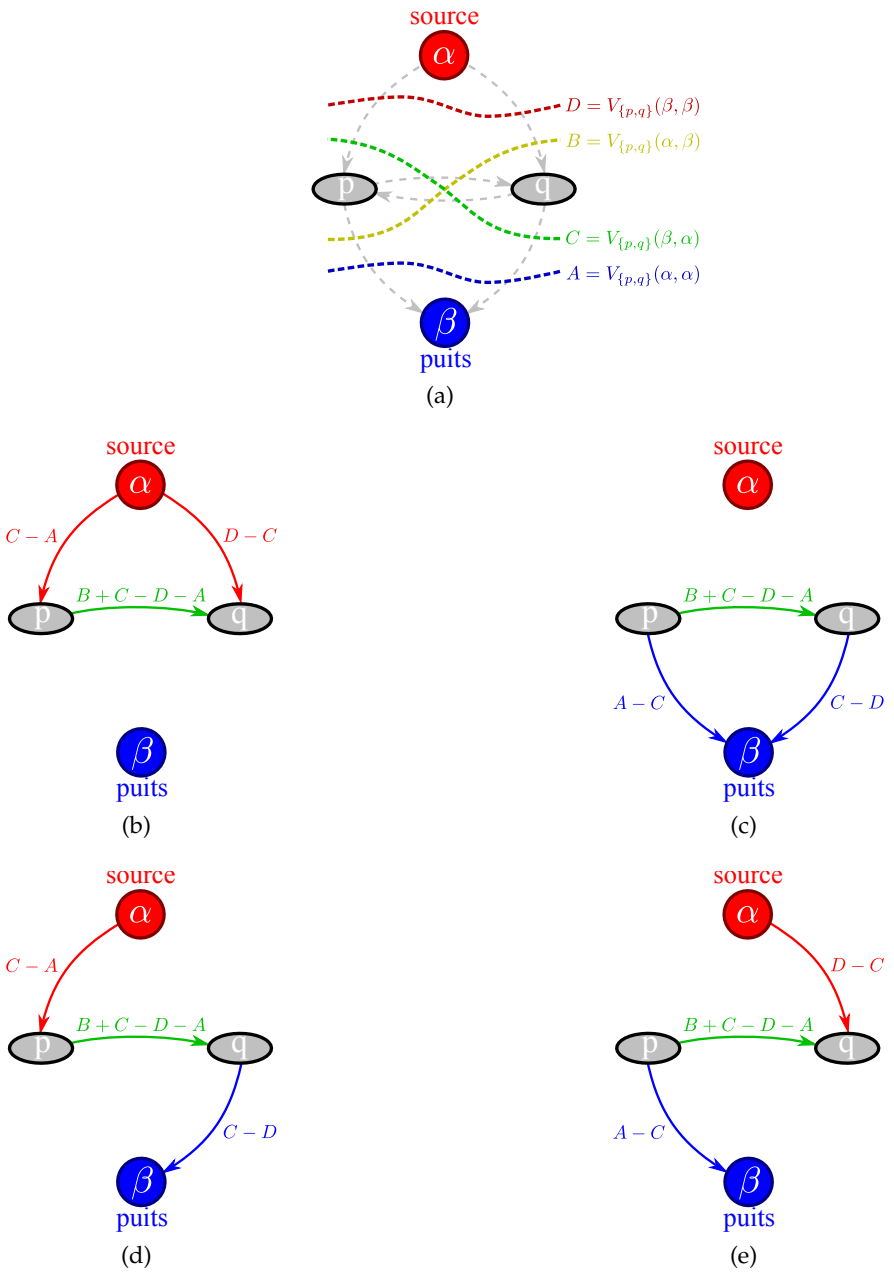


FIGURE 6.13 – (a) Différentes coupes possibles dans un sous-graphe de résolution lié au terme de voisinage dans une itération d'$\{\alpha, \beta\}$-*swap* associée à la minimisation d'une énergie du type de l'équation 6.12 (introduction d'une nouvelle notation pour le poids des coupes). (b) à (e) Configurations possibles du sous-graphe optimisé dans une itération d'$\{\alpha, \beta\}$-*swap* associée à la minimisation d'une énergie du type de l'équation 6.12 avec : (b) $C - A \geq 0$ et $D - C \geq 0$, (c) $A - C \geq 0$ et $C - D \geq 0$, (d) $C - A \geq 0$ et $C - D \geq 0$, (e) $A - C \geq 0$ et $D - C \geq 0$.

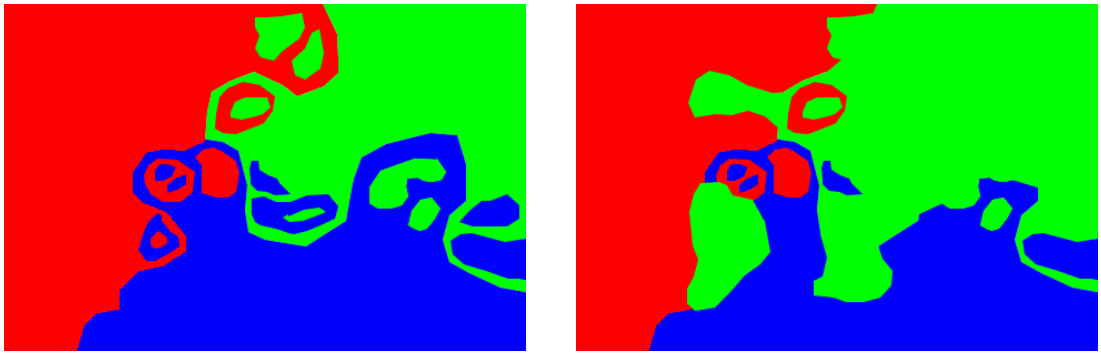
Comme pour le cas binaire, cette construction de graphe à l'avantage de posséder moins d'arcs que la construction d'$\{\alpha,\beta\}$-*swap* "classique" et donc de permettre une recherche de coupe minimale plus rapide.

6.2.2.2 α-*expansion*

Soit $\mathcal{P}_{\overline{\alpha}} = \cup_{\ell\in\{\mathcal{L}\setminus\alpha\}}\mathcal{P}_{\ell(\alpha)}$ l'ensemble des sites qui ne sont pas étiquetés α mais qui sont susceptibles de l'être. Par définition, nous savons donc que, dans le graphe G correspondant, chaque site $p \in \mathcal{P}_{\overline{\alpha}}$ est relié au terminal α et à au moins un autre terminal.

En partant d'un étiquetage quelconque $x \in \mathcal{X}$, une α-*expansion* consiste, pour une étiquette $\alpha \in \mathcal{L}$, à modifier l'étiquetage de certains sites $p \in \mathcal{P}_{\overline{\alpha}}$ en les étiquetant α.

Une illustration de modification d'étiquetage par α-*expansion* se trouve dans la figure 6.14.



(a) (b)

FIGURE 6.14 – (a) Etiquetage des pixels d'une image avec trois étiquettes (*rouge*, *verte* et *bleue*). (b) Etiquetage des pixels après une opération de *verte-expansion*.

Notre objectif étant de minimiser une énergie, nous chercherons donc à obtenir, parmi les configurations qu'il est possible d'atteindre par cet échange d'étiquettes, celle qui minimise le plus l'énergie.

Le graphe orienté $G_{\overline{\alpha}} = (S_{\overline{\alpha}}, A_{\overline{\alpha}})$ pondéré sur les arcs correspondant à une α-*expansion* est construit de la façon suivante :

- $S_{\overline{\alpha}} = \mathcal{P}_{\overline{\alpha}} \cup \left\{ \cup_{\substack{p,q\in\mathcal{P}_{\overline{\alpha}} \\ \{p,q\}\in\mathcal{V} \\ x_p\neq x_q}} s_{\{p,q\}} \right\} \cup \{\alpha,\overline{\alpha}\}$:
  - un sommet pour chaque site de $\mathcal{P}_{\overline{\alpha}}$,
  - un sommet $s_{\{p,q\}}$ pour chaque paire de sites voisins de $\mathcal{P}_{\overline{\alpha}}$ ayant des étiquettes différentes,
  - un sommet source α et un sommet puits $\overline{\alpha}$;
- $A_{\overline{\alpha}} = \left\{ \cup_{\substack{p,q\in\mathcal{P}_{\overline{\alpha}} \\ \{p,q\}\in\mathcal{V} \\ x_p=x_q}} \{(p,q)\cup(q,p)\} \right\} \cup \left\{ \cup_{\substack{p,q\in\mathcal{P}_{\overline{\alpha}} \\ \{p,q\}\in\mathcal{V} \\ x_p\neq x_q}} \{(p,s_{\{p,q\}})\cup(s_{\{p,q\}},p)\cup(q,s_{\{p,q\}})\cup (s_{\{p,q\}},p)\cup(\alpha,s_{\{p,q\}})\} \right\} \cup \left\{ \cup_{p\in\mathcal{P}_{\overline{\alpha}}} \{(\alpha,p)\cup(p,\overline{\alpha})\} \right\}$:
  - des arcs symétriques, appelées v-arcs, reliant chaque paire de sites $p,q \in \mathcal{P}_{\overline{\alpha}}$ qui sont voisins (i.e. $\{p,q\} \in \mathcal{V}$) si $x_p = x_q$:
    - le v-arc (p,q), pondéré par $V_{\{p,q\}}(\alpha, x_q)$, et
    - le v-arc (q,p), pondéré par $V_{\{p,q\}}(x_p,\alpha)$;

ou bien les sites $p, q \in \mathcal{P}_{\overline{\alpha}}$ avec le sommet supplémentaire $s_{\{p,q\}}$ si $x_p \neq x_q$:
  - les v-arcs $(p, s_{\{p,q\}})$ et$(s_{\{p,q\}}, q)$, pondérés par $V_{\{p,q\}}(\alpha, x_q)$, et
  - les v-arcs $(q, s_{\{p,q\}})$ et$(s_{\{p,q\}}, p)$, pondérés par $V_{\{p,q\}}(x_p, \alpha)$;
- des arcs, appelées t-arcs, reliant :
  - la source α à chaque sommet $p \in \mathcal{P}_{\overline{\alpha}}$, pondéré par $D_p(x_p) + \sum_{\substack{q \in \mathcal{V}_p \\ q \notin \mathcal{P}_{\overline{\alpha}}}} V_{\{p,q\}}(x_p, x_q)$,
  - la source α à chaque sommet $s_{\{p,q\}}$, pondéré par $V_{\{p,q\}}(x_p, x_q)$,
  - chaque sommet $p \in \mathcal{P}_{\overline{\alpha}}$ au puits $\overline{\alpha}$, pondéré par $D_p(\alpha) + \sum_{\substack{q \in \mathcal{V}_p \\ q \notin \mathcal{P}_{\overline{\alpha}}}} V_{\{p,q\}}(\alpha, x_q)$.

La construction du graphe $G_{\overline{\alpha}}$ est donc très similaire à celle du graphe $G_{\alpha\beta}$, à l'exception des sommets $s_{\{p,q\}}$ à rajouter "entre" les sites de $\mathcal{P}_{\overline{\alpha}}$ qui sont voisins et qui ont des étiquettes différentes. Ces sommets supplémentaire sont ajoutés afin de pouvoir représenter le coût du changement d'étiquette entre les deux sites au travers de poids l'arc qui lui arrive depuis la source α.

Une fois une coupe minimale C calculée dans le graphe $G_{\overline{\alpha}} = (S_{\overline{\alpha}}, A_{\overline{\alpha}})$, chaque site $p \in \mathcal{P}_{\overline{\alpha}}$ du graphe $G'_{\overline{\alpha}} = (S_{\overline{\alpha}}, A_{\overline{\alpha}} \setminus C)$ n'est plus connecté qu'à la source α ou au puits $\overline{\alpha}$ puisque la coupe contient un seul de ces deux arcs. En conséquence de quoi on déduit le résultat du problème d'étiquetage comme suit :
- si le site p est connecté à la source α, alors p est étiqueté α (i.e. $x_p = \alpha$) ;
- si le site p est connecté au puits $\overline{\alpha}$, alors l'étiquette de p demeure inchangée.

Remarque 6.11.
Il est à noter que la construction de graphe présentée ici diffère sensiblement des cas présentés dans [36, 35, 38]. En effet, outre les différences déjà mentionnées dans la remarque 6.9, dans ces articles, la construction inclut les sites déjà étiquetés auxquels un t-arc de poids infini est ajouté afin de s'assurer qu'ils ne changent pas d'étiquette. La construction présentée ici permet ainsi de réduire le nombre de sommets et d'arcs dans le graphe, accélérant ainsi la recherche d'une coupe minimale.
Ceci rejoint la construction présentée ci-avant pour les $\{\alpha, \beta\}$-swaps et s'explique par les équivalences de constructions de graphes présentées dans la section 5.5.

En 1999, il fut mis en évidence dans [36, 38] que, dans le cas général d'une énergie du type de l'équation 6.12, une condition suffisante pour permettre la minimisation par *α-expansion* est que $V_{\{p,q\}}$ soit une métrique sur $\mathcal{L}$. Nous en rappelons ci-dessous la définition.

La fonction $V_{\{p,q\}}$ est une **métrique** sur l'espace des étiquettes $\mathcal{L}$ si $\forall \alpha, \beta, \gamma \in \mathcal{L}_p \cap \mathcal{L}_q$:

$$\begin{aligned} &V_{\{p,q\}}(\alpha, \beta) = 0 \Leftrightarrow \alpha = \beta \\ &V_{\{p,q\}}(\alpha, \beta) = V_{\{p,q\}}(\beta, \alpha) \geq 0 \\ &V_{\{p,q\}}(\alpha, \beta) \leq V_{\{p,q\}}(\alpha, \gamma) + V_{(p,q)}(\gamma, \beta) \end{aligned} \tag{6.19}$$

Cette condition, plus restrictive que celle de semi-métrique (voir équation 6.17), permet donc d'utiliser les *α-expansions* pour la minimisation de certaines énergies liées à un problème du type de celui de la sous-section 6.1.2. La nécessité de la condition d'inégalité triangulaire sera vue un peu plus loin.

Une illustration des différentes coupes possibles sur un sous-graphe induit par la source α, le puits $\overline{\alpha}$ et deux sites $p, q \in \mathcal{P}_{\overline{\alpha}}$ voisins (i.e. $\{p, q\} \in \mathcal{V}$) tels que $x_p = x_q$ se trouve dans la figure 6.15.

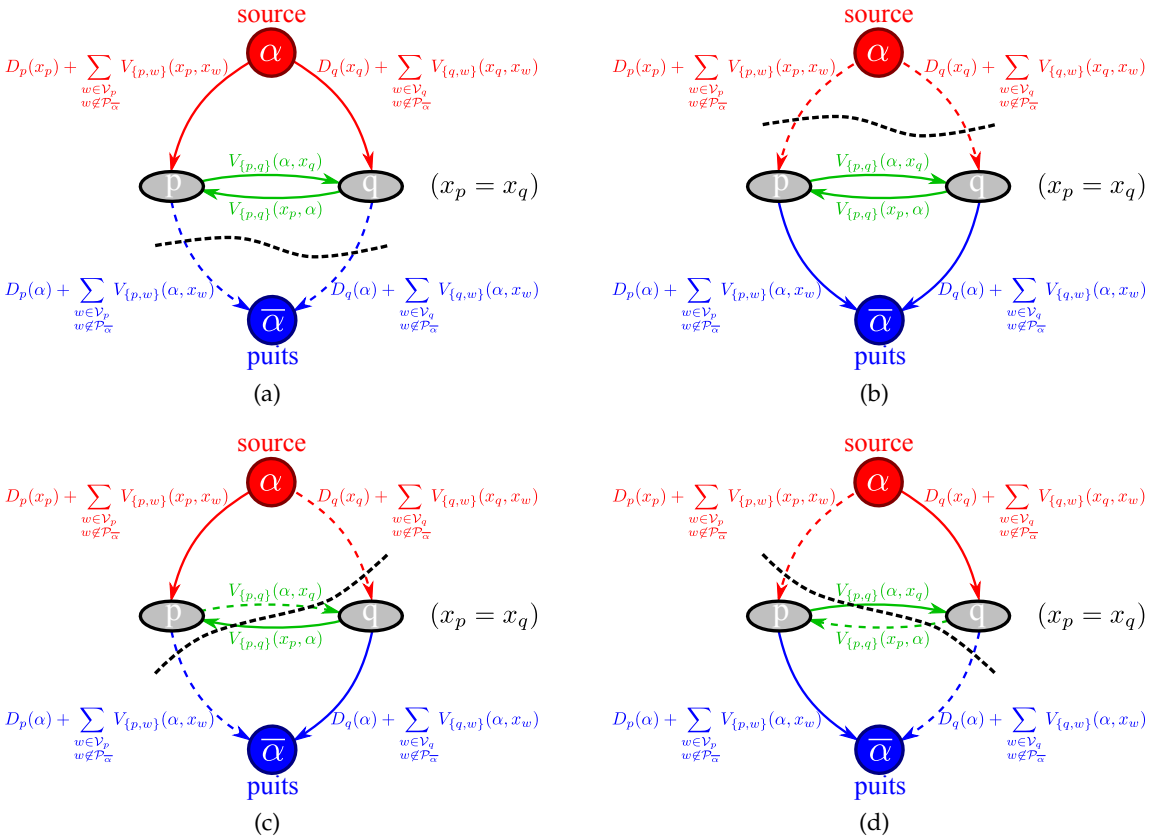


FIGURE 6.15 – Configurations possibles de coupes du sous-graphe induit par la source α, le puits $\overline{\alpha}$ et deux sites voisins $p, q \in \mathcal{P}_{\alpha\beta}$ tels que $x_p = x_q$ dans une itération d'*α-expansion* associée à la minimisation d'une énergie du type de l'équation 6.12.

Une illustration des différentes coupes possibles sur un sous-graphe induit par la source α, le puits $\overline{\alpha}$, deux sites $p, q \in \mathcal{P}_{\overline{\alpha}}$ voisins (i.e. $\{p,q\} \in \mathcal{V}$) tels que $x_p \neq x_q$ et leur sommet supplémentaire $s_{\{p,q\}}$ se trouve dans la figure 6.16.

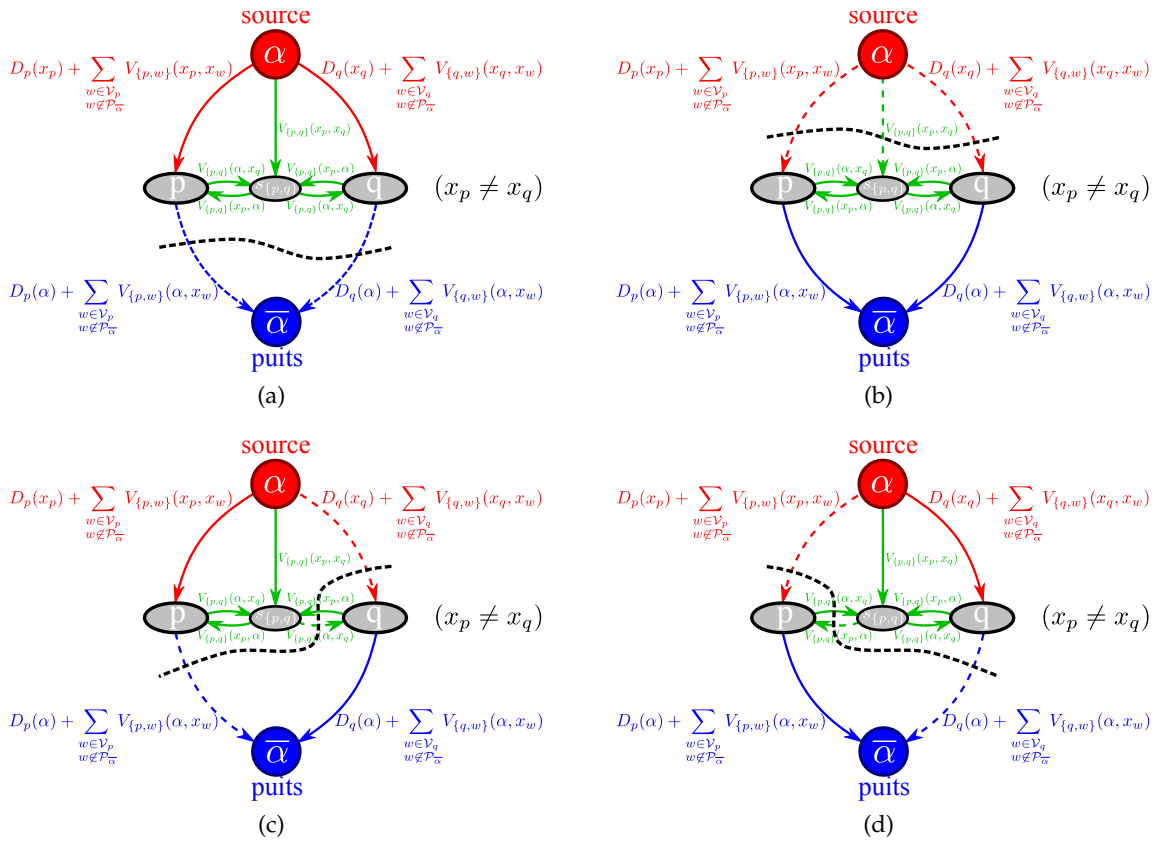


FIGURE 6.16 – Configurations possibles de coupes du sous-graphe induit par la source α, le puits $\overline{\alpha}$, deux sites voisins $p, q \in \mathcal{P}_{\alpha\beta}$ tels que $x_p \neq x_q$ et leur sommet supplémentaire $s_{\{p,q\}}$ dans une itération d'*α-expansion* associée à la minimisation d'une énergie du type de l'équation 6.12.

Il est à noter que certaines coupes du sous-graphe montré dans la figure 6.16 ne peuvent être des coupes minimales. C'est le cas de celles présentées dans la figure 6.17 :

- pour les coupes des figures 6.17a et 6.17b, par construction du graphe, celles des figures 6.16c et, respectivement, 6.16d s'avèrent de poids inférieur ;
- pour la coupe de la figure 6.17c, suite à l'inégalité triangulaire de la condition de métrique sur $V_{\{p,q\}}$, il s'avère que la coupe de la figure 6.16b s'avère également de poids inférieur.

Une illustration de problème d'étiquetage avec une itération d'*α-expansion* par coupe minimale se trouve dans la figure 6.18.

Chaque exécution d'*α-expansion* tente donc de modifier l'étiquetage d'entrée $x \in \mathcal{X}$ en un étiquetage $x' \in \mathcal{X}$ de sorte à en diminuer l'énergie correspondante ($E(x') \leq E(x)$).

Afin d'obtenir la meilleure approximation possible du minimum global d'un problème d'étiquetage par *α-expansions*, il suffit de procéder comme suit :

1. Initialiser les sites avec un étiquetage quelconque $x \in \mathcal{X}$

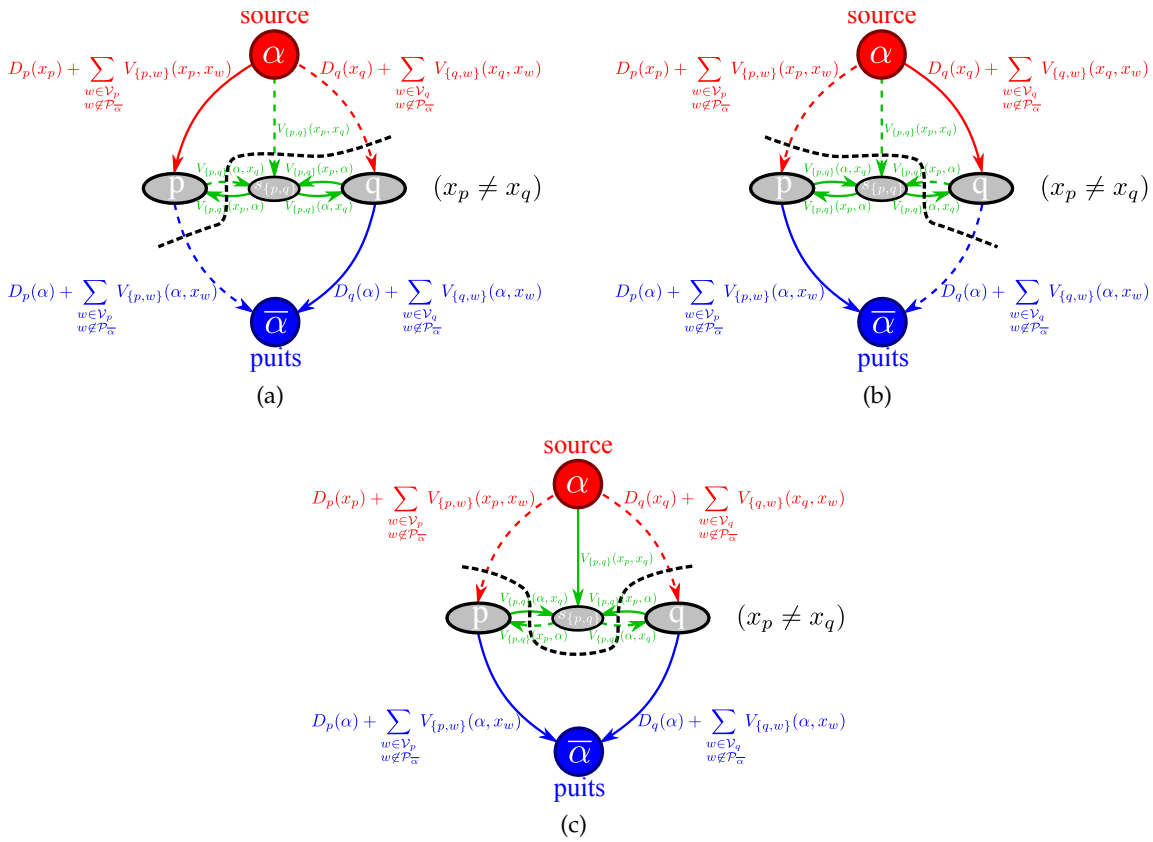


FIGURE 6.17 – Configurations impossibles de coupes du sous-graphe induit par la source α, le puits $\overline{\alpha}$, deux sites voisins $p, q \in \mathcal{P}_{\alpha\beta}$ tels que $x_p \neq x_q$ et leur sommet supplémentaire $s_{\{p,q\}}$ dans une itération d'*α-expansion* associée à la minimisation d'une énergie du type de l'équation 6.12.

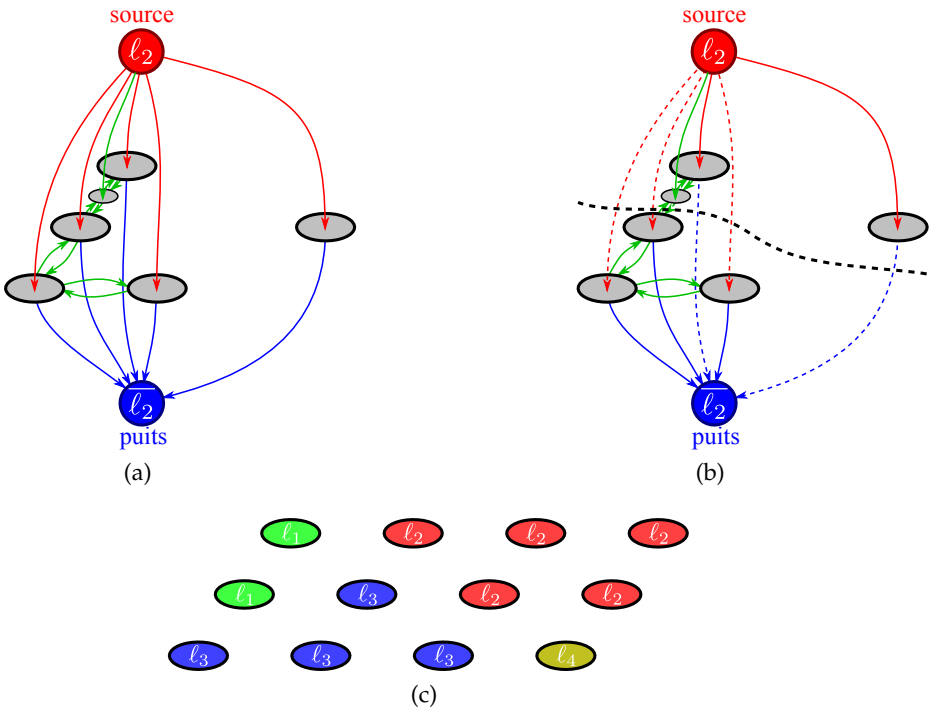


FIGURE 6.18 – (a) Graphe orienté $G_{\overline{rouge}}$ pour le problème représenté dans la figure 6.8a et d'après l'étiquetage de la figure 6.8c. (b) Coupe minimale C sur le graphe $G_{\overline{rouge}}$ (la ligne pointillée traversant les arcs pointillés). (c) Résultat du problème d'étiquetage correspondant à la coupe C.

2. Initialiser $succès = 0$
3. Pour toute étiquette $\alpha \in \mathcal{L}$:
   3-1 Chercher $\check{x} = \arg\min\{E(x')\}$ où $E(x')$ est une configuration obtenue par une *α-expansion* sur x
   3-2 Si $E(\check{x}) < E(x)$, alors $x = \check{x}$ et $succès = 1$
4. Si $succès = 1$, retourner à l'étape 2
5. Renvoyer x

Dans cet algorithme également, on appelle **itération** une unique exécution des étapes 3.1-3.2 et **cycle** une exécution des étapes 2-4.

Un cycle comporte n itérations (avec $n = |\mathcal{L}|$).

L'ordre d'exécution des itérations pour chaque étiquette dans un cycle n'a pas d'importance. Un cycle est considéré comme "réussi" si au moins une itération a permis de trouver un étiquetage strictement meilleur. L'algorithme s'arrête dès qu'un cycle n'a permis aucune amélioration et que l'étiquetage est stabilisé. Cet algorithme garantit de terminer en un nombre fini de cycles. En pratique, seuls quelques cycles sont nécessaires pour arriver à stabilité mais la majeure partie des changements d'étiquettes se produisent durant le premier cycle.

Il a été mis en évidence dans [38] que l'approximation offerte par les *α-expansions* est bornée comme le montre le théorème suivant.

Théorème 6.12 ([38], théorème 6.1).
Soit

$$c = \frac{\max\{V_{\{p,q\}}(x_p, x_q) \mid \{p,q\} \in \mathcal{V}, x_p \neq x_q\}}{\min\{V_{\{p,q\}}(x_p, x_q) \mid \{p,q\} \in \mathcal{V}, x_p \neq x_q\}}.$$

Si x est le minimum local obtenu comme résultat d'α-expansions et x^* le minimum global, alors $E(x) \leq 2c \times E(x^*)$.

Tout comme pour le cas des $\{\alpha, \beta\}$-*swaps*, on déduit du théorème 6.4, mis en évidence en 2002 dans [123], une nouvelle condition sur $V_{\{p,q\}}$, plus large que celle de métrique, nécessaire et suffisante à la minimisation par α-*expansions* d'une énergie du type de l'équation 6.12, présentée dans le corollaire 6.13.

Corollaire 6.13.
Un problème d'étiquetage peut être résolu pour l'énergie $E(x)$ (voir équation 6.12) à l'aide d'α-expansions si et seulement si la fonction $V_{\{p,q\}}$, représentant le terme de voisinage, respecte la ***condition de régularité*** *suivante pour tout $\alpha, \beta, \gamma \in \mathcal{L}_p \cap \mathcal{L}_q$:*

$$V_{\{p,q\}}(\alpha, \alpha) + V_{\{p,q\}}(\beta, \gamma) \leq V_{\{p,q\}}(\alpha, \gamma) + V_{\{p,q\}}(\beta, \alpha) \tag{6.20}$$

Il est à noter que si $V_{\{p,q\}}$ est une métrique, alors $V_{\{p,q\}}$ respecte la condition de régularité du corollaire 6.13. Néanmoins, cette condition est plus large que celle de métrique, permettant, là encore, la minimisation de problèmes d'étiquetage avec une énergie anisotrope.

La construction de graphe optimisée valable pour toute énergie ayant un terme de voisinage régulier présentée dans [123] est également applicable au cas de l'α-*expansion* (voir en fin de section 6.2.1). Sa construction se fait comme précédemment, excepté que l'on ne considère plus que les sites de $\mathcal{P}_{\overline{\alpha}}$ au lieu de tous les sites de $\mathcal{P}$.

Comme pour le cas de l'$\{\alpha, \beta\}$-*swap*, la construction des graphes pour une itération d'α-*expansion*, avant fusion, pour chacun des termes d'une énergie à terme de voisinage régulier du type de l'équation 6.12 se fait de manière très similaire à précédemment. Nous ne donnerons donc pas davantage de détails et nous contenterons d'illustrer les différentes configurations de graphes possibles la figure 6.19 pour le terme direct et dans la figure 6.20 pour le terme de voisinage.

Encore plus qu'avec le cas binaire ou un $\{\alpha, \beta\}$-*swap*, cette construction de graphe à l'avantage de posséder moins d'arcs que la construction d'une α-*expansion* "classique" puisqu'elle ne nécessite aucun ajout de noeud supplémentaire pour représenter une différence d'étiquettes entre deux sites voisins. Avec moins de noeuds et moins d'arcs, la recherche de coupe minimale sur ce type de graphe est bien plus rapide.

Bien que la minimisation par α-*expansions* offre des résultats légèrement moins bons que celle par $\{\alpha, \beta\}$-*swaps*, la minimisation par α-*expansions* est plus rapide à exécuter : n itérations par cycle au lieu de $\frac{n^2-n}{2}$. De plus, les α-*expansions* convergent vers le résultat final en moins de cycles que les $\{\alpha, \beta\}$-*swaps*. Des temps trois fois plus rapides avec α-*expansions* qu'avec $\{\alpha, \beta\}$-*swaps* sont indiqués dans [38] (qui ne prenait alors pas en compte la construction de graphe optimisée présentée ci-avant).

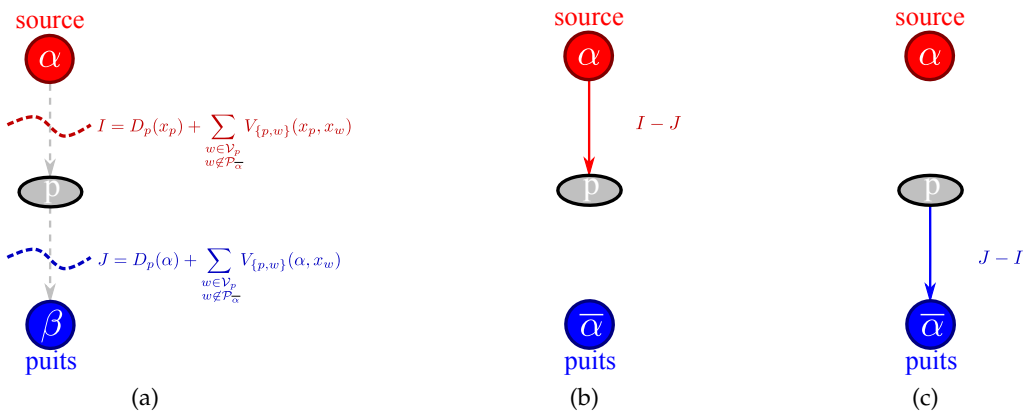


FIGURE 6.19 – (a) Différentes coupes possibles dans un sous-graphe de résolution lié au terme direct dans une itération d'*α-expansion* associée à la minimisation d'une énergie du type de l'équation 6.12 (introduction d'une nouvelle notation pour le poids des coupes). (b,c) Configurations possibles du sous-graphe optimisé dans une itération d'*α-expansion* associée à la minimisation d'une énergie du type de l'équation 6.12 avec : (b) $I - J \geq 0$, (c) $J - I \geq 0$.

Par conséquent, quand les conditions le permettent, puisque celles pour les *α-expansions* sont plus restrictives que celles pour les $\{\alpha, \beta\}$-*swaps*, les *α-expansions* sont souvent préférées. C'est pourquoi cette méthode est si répandue en vision par ordinateur.

6.2.3 Autres cas particuliers

Dans cette section sont présentés des cas particuliers de minimisation par coupe de graphe.

6.2.3.1 Cas multi-level

On considère ici un problème d'étiquetage à n étiquettes associé à une énergie du type de l'équation 6.12 tel que :

- l'ensemble d'étiquettes $\mathcal{L}$ soit isomorphe à $\{1, 2, \ldots, n\}$ (pour des raisons de simplicité nous considérerons par la suite que $\mathcal{L} = \{1, 2, \ldots, n\}$) ;
- le terme de voisinage soit de la forme :

$$V_{\{p,q\}}(x_p, x_q) = \lambda_{\{p,q\}} |x_p - x_q| \tag{6.21}$$

Un tel problème peut être résolu de manière optimale avec une seule coupe minimale à chercher sur le graphe $G = (S, A)$ construit comme suit :

- $S = \mathcal{P}^{n-1} \cup \{s, t\}$:
  - pour chaque site $p \in \mathcal{P}$, $(n-1)$ sommets $p_1, \ldots, p_{n-1}$,
  - un sommet source s et un sommet puits t ;
- $A = \left\{ \cup_{p \in \mathcal{P}} \left\{ \cup_{i=1}^{n-2} (p_i, p_{i+1}) \cup (s, p_1) \cup (p_{n-1}, t) \right\} \right\} \cup \left\{ \cup_{\substack{p,q \in \mathcal{P} \\ \{p,q\} \in \mathcal{V}}} \left\{ \cup_{i=1}^{n-1} \{(p_i, q_i) \cup (q_i, p_i)\} \right\} \right\}$:
  - pour chaque site $p \in \mathcal{P}$:
    - l'arc (s, p_1), pondéré par $D_p(1)$;

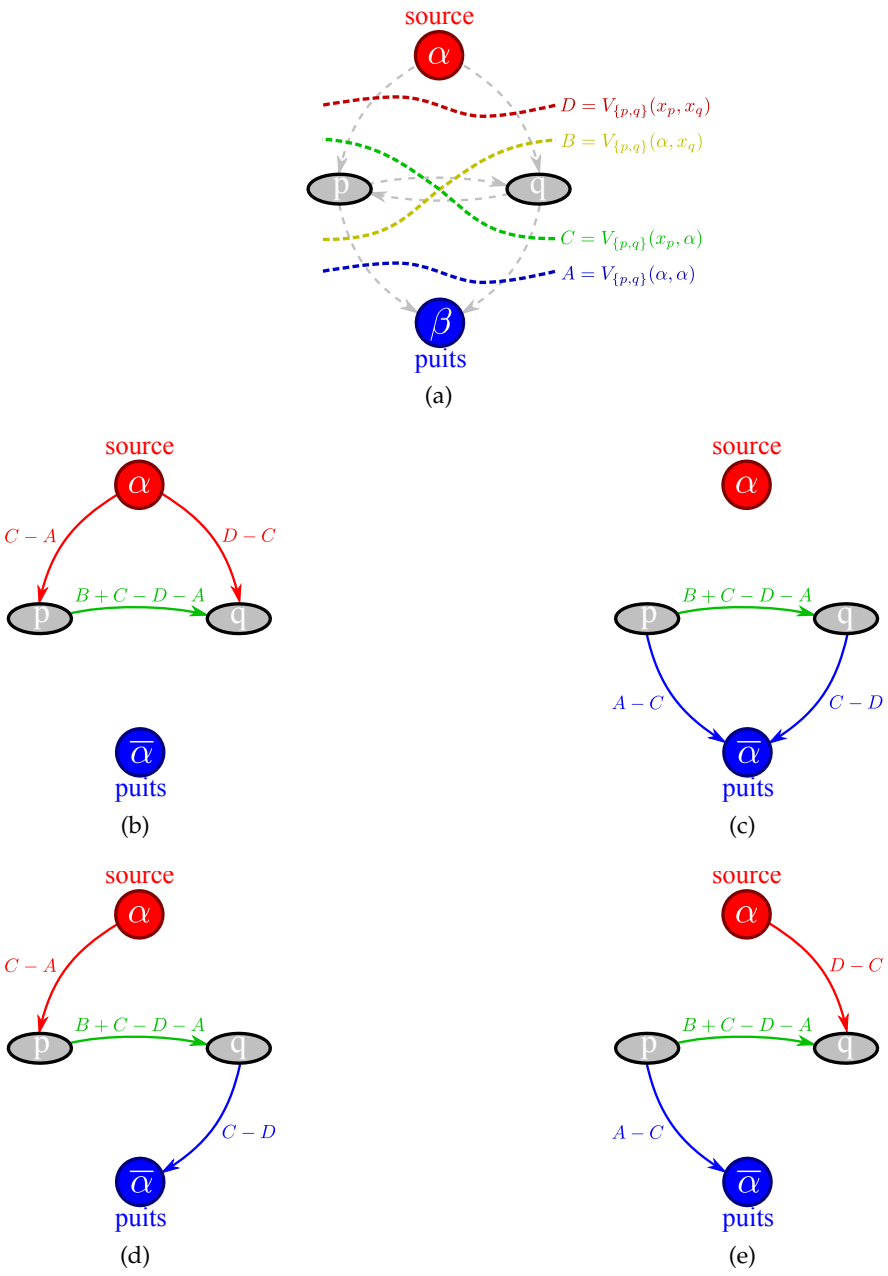


FIGURE 6.20 – (a) Différentes coupes possibles dans un sous-graphe de résolution lié au terme de voisinage dans une itération d'*α-expansion* associée à la minimisation d'une énergie du type de l'équation 6.12 (introduction d'une nouvelle notation pour le poids des coupes). (b) à (e) Configurations possibles du sous-graphe optimisé dans une itération d'*α-expansion* associée à la minimisation d'une énergie du type de l'équation 6.12 avec : (b) $C - A \geq 0$ et $D - C \geq 0$, (c) $A - C \geq 0$ et $C - D \geq 0$, (d) $C - A \geq 0$ et $C - D \geq 0$, (e) $A - C \geq 0$ et $D - C \geq 0$.

  - pour tout $i \in [1; n-2]$, l'arc (p_i, p_{i+1}), pondéré par $D_p(i+1)$;
  - l'arc (p_{n-1}, t), pondéré par $D_p(n)$;
- pour toute paire de sites $p, q \in \mathcal{P}$ qui sont voisins (i.e. $\{p, q\} \in \mathcal{V}$) et pour tout $i \in [1; n-1]$, les arcs symétriques (p_i, q_i) et (q_i, p_i) pondérés par $\lambda_{\{p,q\}}$.

Il est à noter que la différence majeure entre ce cas et le cas général est que chaque variable n'est plus représentée par un unique sommet mais par une "colonne" de $n-1$ sommets.

Une fois une coupe minimale C calculée dans le graphe $G = (S, A)$, pour chaque site $p \in \mathcal{P}$, on regarde dans la colonne de sommets lui correspondant où passe la coupe C. L'arc qu'elle coupe dans cette colonne représente l'étiquette à attribuer à p. En conséquence de quoi on déduit le résultat du problème d'étiquetage comme suit :

- si le sommet p_1 est dans la composante connexe contenant le puits t, alors p est étiqueté 1 (i.e. $x_p = 1$) ;
- si le sommet p_{n-1} est dans la composante connexe contenant la sources s, alors p est étiqueté n (i.e. $x_p = n$) ;
- si il existe $i \in [1; n-2]$ tel que le sommet p_i soit dans la composante connexe contenant la sources s et le sommet p_{i+1} soit dans la composante connexe contenant le puits t, alors p est étiqueté $(i+1)$ (i.e. $x_p = i+1$).

Une illustration de problème d'étiquetage avec une résolution par graphe multi-level se trouve dans la figure 6.21.

La construction de ce type de graphe pour la minimisation d'énergie fut proposée en 1998 dans [35].

On dit qu'une coupe C du graphe $G = (S, A)$ construit comme expliqué ci-dessus est **réalisable** si, pour tout site $p \in \mathcal{P}$ un seul arc de $\{(s, p_1), \cup_{i=1}^{n-2}(p_i, p_{i+1}), (p_{n+1}, t)\}$ appartient à la coupe C.

Il est à noter que, selon la pondération des arcs liée à l'énergie, avec cette construction de graphe, une coupe n'est pas nécessairement réalisable, rendant par la même occasion la transcription en terme d'étiquetage impossible (voir figure 6.21c). C'est pourquoi, la même année, Hiroshi Ishikawa et Davi Geiger [108], en se basant sur les travaux de Sébastien Roy and Ingemar J. Cox [180], proposèrent une variante de cette construction qui consiste à rajouter des arcs infinis comme suit : pour tout site $p \in \mathcal{P}$, pour tout $i \in [1; n-2]$, ajouter un arc (p_{i+1}, p_i) de poids infini.

Une illustration de ce nouveau type de construction de graphe se trouve dans la figure 6.22.

Hiroshi Ishikawa généralisa cette méthode de résolution à toute fonction convexe en 2003 [107] en rajoutant de nombreux arcs au graphe décrit précédemment. Ceci eut pour conséquence de rendre le calcul d'une coupe minimale sur ce graphe extrêmement coûteux en temps et en espace mémoire. Il fut cependant mis en évidence dans [184] que, dans ce cas, les frontières obtenues sont potentiellement plus lissées qu'il ne faudrait puisque la coupe est de moindre poids lorsqu'elle passe par plusieurs "petits" changements de niveau que par un seul "grand". Il n'y a donc plus préservation de la discontinuité.

Il est à souligner que cette méthode demeure néanmoins le seul cas connu où un MRF avec plus de deux étiquettes peut être résolu de façon optimale.

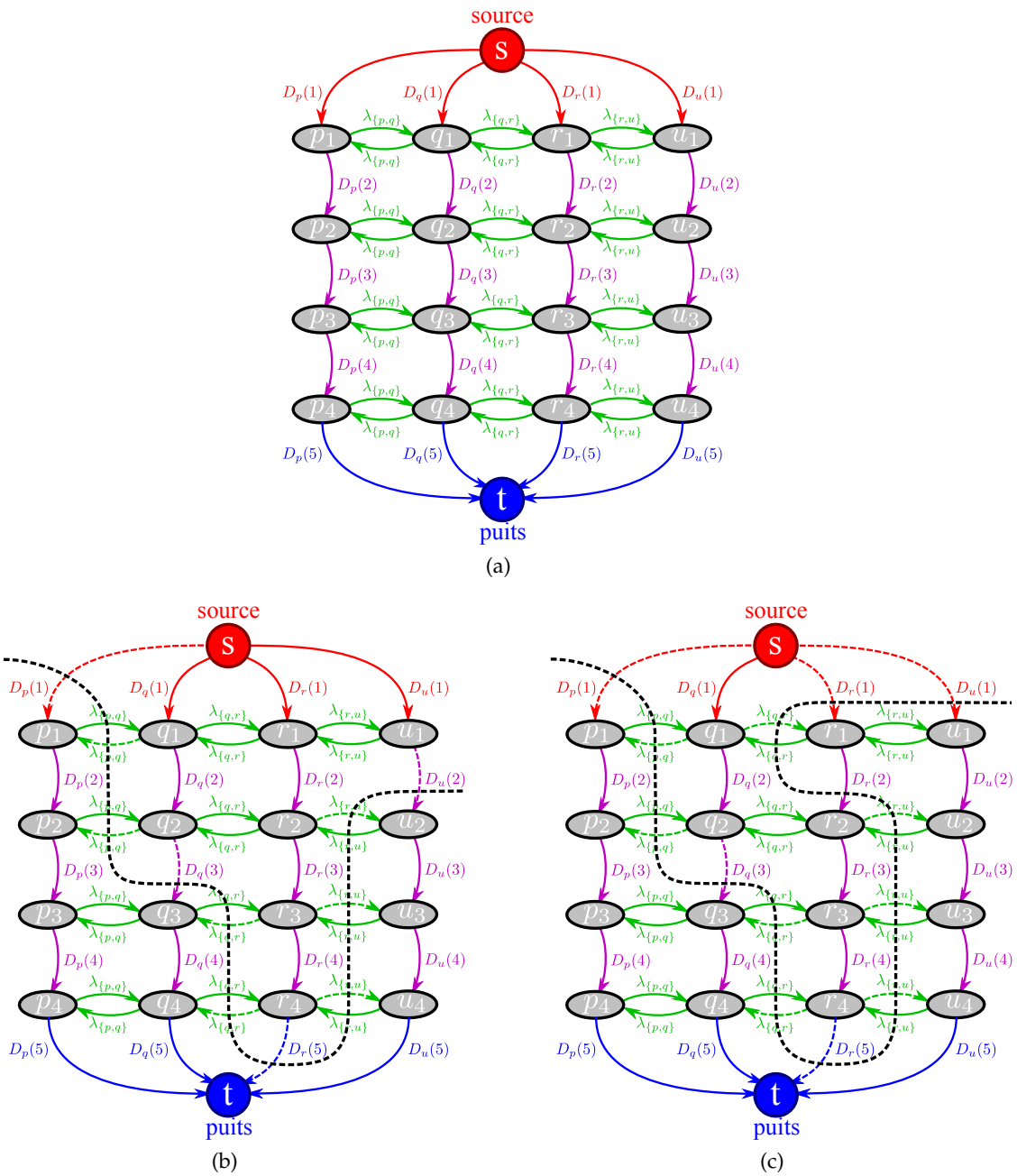


FIGURE 6.21 – (a) Graphe multi-level G pour la minimisation d'une énergie du type de l'équation 6.12 avec un terme de voisinage du type de l'équation 6.21. (b) Coupe réalisable sur G. (c) Coupe non-réalisable sur G.

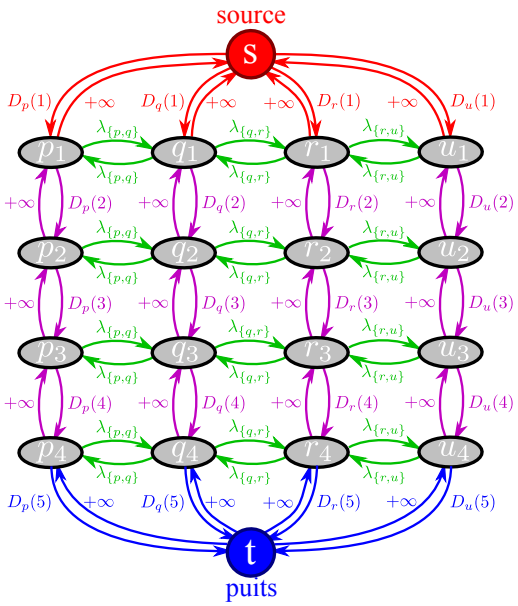


FIGURE 6.22 – Graphe multi-level avec arcs infinis pour la minimisation d'une énergie du type de l'équation 6.12 avec un terme de voisinage du type de l'équation 6.21.

6.2.3.2 Modèle de Potts

Comme nous l'avons vu dans la sous-section 6.1.3.2, le terme de voisinage dépend uniquement du fait que les étiquettes de deux sites voisins soient identiques ou non (et des sites voisins concernés dans le cas du modèle de Potts généralisé). Les étiquettes en elles-mêmes importent peu, seul le fait qu'elles soient différentes ou non entre en jeu.

Il fut montré dans [35] que, dans ce cas, on peut se ramener à une recherche *multiway cut* de poids minimal sur un graphe non-orienté construit comme présenté dans la section 6.2.2 auquel on ajoute une pondération. Cependant, sauf pondération bien choisie, la recherche d'une *multiway cut* de poids minimal par l'heuristique d'isolation présentée dans la sous-section 5.6.3 ne garantit pas que la *multiway cut* obtenue soit réalisable. C'est pourquoi, même dans ce cas, mieux vaut utiliser la méthode de minimisation par $\{\alpha, \beta\}$-*swaps* ou par α-*expansions*.

6.3 Autres méthodes de résolution

Comme nous l'avons déjà vu dans la sous-section 6.1.2, la résolution d'un problème d'étiquetage par minimisation d'énergie s'avère NP-complet. Nous allons présenter brièvement ci-après certaines méthodes parmi les plus couramment utilisées en vision par ordinateur jusqu'au développement des méthodes de minimisation par coupes de graphes ainsi que de nouvelles inspirées par ces dernières.

6.3.1 Recuit simulé

Le recuit simulé (*simulated annealing* en anglais) [117] est une méthode issue de l'analogie entre l'énergie à minimiser et la température d'un système physique que l'on souhaite faire baisser. Cette minimisation se résout avec une méthode classique de descente de gradient. Néanmoins, la température est libre de pouvoir remonter (de façon contrôlée) de sorte à pouvoir sortir d'un éventuel minimum local dans lequel elle se serait retrouvée. C'est ce qu'on appelle un recuit.

Cette méthode fut introduite en traitement d'images par Stuart et Donald Geman en

1984 [90].

Afin d'obtenir de bons résultats, la diminution de l'énergie doit se faire très lentement au travers de nombreux calculs. C'est pourquoi, généralement, on se contente d'un compromis entre une diminution plus rapide de l'énergie, et donc un temps de calcul moindre, et le risque de ne pouvoir sortir d'un minimum local.

Il est à noter que, d'après D. M. Greig, B. T. Porteous et A. H. Seheult [102], la minimisation d'énergie par recuit simulé perd en efficacité pour des termes de voisinages élevés.

6.3.2 *Iterated conditional modes*

Les ***iterated conditional modes*** (**ICM**), développés par Julian E. Besag en 1986 [28], sont une méthode qui consiste à maximiser les probabilités conditionnelles locales de façon séquentielle grace à un algorithme glouton : à chaque itération, on détermine, pour chaque site, l'étiquette minimisant l'erreur de classification en fonction des étiquettes présumées de ses voisins.

Il est à noter que cette méthode est très sensible à l'initialisation et peut se retrouver bloquer dans un minimum local.

6.3.3 Ensembles de niveaux

Les ensembles de niveaux (*level sets* en anglais), développés dans les années 1980 par Stanley Osher et James A. Sethian [160], sont une méthode consistant à représenter une courbe ϕ par le niveau zéro d'une fonction d'ensemble de niveaux Φ (i.e. $\phi = \Phi(0)$). Il en découle que faire évoluer la fonction d'ensemble de niveaux Φ fait également évoluer la courbe ϕ. Modifier Φ est certes plus coûteux en temps et en espace mémoire puisqu'il y a une dimension supplémentaire au problème initial, mais cela permet de travailler dans un contexte implicite et intrinsèque. La minimisation se fait par des équations aux dérivées partielles (EDP).

Voir également [40, 187, 188, 189].

Malgré les optimisations apportées à cette méthode (voir les sous-sections 6.3.3.1 et 6.3.3.2),les ensembles de niveaux demeurent malgré tout coûteux en temps calcul et ont le risque de ne trouver qu'un minimum local. C'est pourquoi, bien qu'ils furent très utilisés dans les années 2000, ils se retrouvent aujourd'hui délaissés en faveur des méthodes de minimisation par graphes.

6.3.3.1 *Narrow bands*

En 1995, David Adalsteinsson et James A. Sethian introduisent dans [2] la méthode de calcul par *narrow bands*. Cela consiste à ne calculer l'évolution de la fonction implicite que dans une bande située de part et d'autre de la courbe de niveau zéro, accélérant ainsi la recherche du minimum.

6.3.3.2 *Fast marching* et chemins minimaux

Les *fast marching* sont une méthode de recherche du minimum par ensemble de niveaux introduite dans [189] plus rapide que celle des *narrow bands* mais restreinte à certaines classes d'EDP. Celle-ci repose sur la recherche de chemins minimaux [50, 51].

Par la suite, ces derniers ont cependant pu êtres utilisés indépendamment pour la recherche de régions, de contours multiples ou encore la complétion de courbes. Le chapitre de Laurent Cohen [49] permet d'avoir un bon aperçu de ses applications.

6.3.4 *Loopy belief propagation*

La ***loopy belief propagation*** (**LBP**) est une méthode d'approximation proposée par Judea Pearl en 1988 dans [165] reposant sur la ***belief propagation*** (**BP**). Ces derniers, proposés par Judea Pearl également six ans plus tôt dans [164], permettent de trouver une solution optimale lorsque le graphe représentant le problème d'étiquetage est un arbre (autrement dit sans cycle) par passage de messages partant des feuilles. Dans sa généralisation à un graphe quelconque (qui peut donc contenir des cycles), les messages sont passés dans un ordre aléatoire sur les sommets. Des optimisations algorithmiques sont présentées dans [84].

Il est à noter que, d'après [196, 197], les résultats obtenus par LBP sont considérés comme moins bons ou équivalents à ceux obtenus par coupes de graphes.

6.3.5 Coupes normalisées

Les coupes normalisées (*normalized cuts* en anglais), introduits par Jianbo Shi et Jitendra Malik en 2000 dans [191], sont une méthode de minimisation inspirée de la segmentation de graphe et du problème des k-coupes minimales (voir sous-section 5.6.1) associée à un critère global mesurant à la fois la dissimilarité globale entre les différentes régions et la similarité dans chaque région.

Le graphe construit est un graphe complet non-orienté (i.e. toute paire de sommets est reliée par une arête - voir sous-section 1.4.6) où chaque arête a un poids d'autant plus grand que les sommets sont considérés comme semblables. La méthode de résolution proposée se fait par la recherche de vecteurs propres sur la matrice associée au graphe.

6.3.6 *Tree-reweighted message passing*

Les ***tree-reweighted message passing*** (**TRW**), développés par Martin J. Wainwright, Tommi S. Jaakkola et Alan S. Willsky en 2003 [207] et améliorés par Vladimir Kolmogorov en 2006 [121] (garantissant alors la convergence), sont une méthode consistant à trouver une famille d'arbres couvrantes (i.e. un ensemble d'arbres tels que chaque arête du graphe appartient à au moins un arbre - à ne pas confondre avec une forêt couvrante) sur chacun desquels on appliquera un algorithme de *belief propagation*.

D'après [122], les résultats obtenus par TRW sont équivalents à ceux obtenus par coupes de graphes lorsque le graphe est faiblement connecté (avec un 4-voisinage par exemple) mais moins bons (voir ne convergent pas du tout) lorsque le graphe est fortement connecté. De plus, les temps de calculs sont 3 à 5 fois plus longs. Toutefois, cette méthode permet de minimiser une plus grande variété d'énergies.

6.3.7 Marches aléatoires

Les marches aléatoires (*random walks* en anglais) furent introduites par Leo Grady en 2006 dans [100]. Elles consistent, en considérant dans un graphe un certain nombre de sommets préalablement étiquetés, à calculer les probabilités qu'un "marcheur aléatoire" partant d'un sommet non étiqueté arrive à chacun des différents sommets étiquetés puis à attribuer l'étiquette correspondant à la forte probabilité. La probabilité de passer d'un sommet à un de ses voisins est représentée par le poids de l'arête correspondante. La résolution de ce problème se fait grâce à l'algèbre linéaire (en l'occurrence, par résolution de systèmes linéaires).

6.3.8 Surfaces minimales

La méthode des surfaces minimales fut introduite par Ben Appleton et Hugues Talbot en 2006 dans [12]. Elle consiste à calculer un flot maximal dans un espace continu grâce à un système d'EDP simulant un fluide idéal avec des contraintes de vitesse isotropes. Une surface minimale peut alors être déduite grâce à une fonction auxiliaire de potentiel.

Cette méthode peut être considérée comme une version continue de l'algorithme de recherche de coupe minimale par *push-relabel* proposé par Andrew V. Goldberg et Robert E. Tarjan [92, 93, 94]. Elle a ainsi l'avantage d'éviter les problèmes liés à la discrétisation des méthodes par coupe de graphe, mais cela se fait au dépend du temps de calcul qui est considérablement accru.

6.3.9 *Logcut*

Les *logcuts* furent introduites par Victor S. Lempitsky, Carsten Rother et Andrew Blake en 2007 dans [135]. Comme pour les $\{\alpha,\beta\}$-*swaps* ou les α-*expansions* (voir section 6.2.2.1 et 6.2.2.2), cette méthode repose sur des recherches successives de coupes minimales binaires sur des graphes. Néanmoins, le nombre de coupes minimales à exécuter croît de façon logarithmique par rapport au nombre d'étiquettes possibles, contrairement aux $\{\alpha,\beta\}$-*swaps* ou aux α-*expansions* dont la croissance est, respectivement, quadratique et linéaire.

En contrepartie, cet algorithme nécessite une étape d'apprentissage très gourmande en espace mémoire.

6.3.10 *Fast primal-dual*

La méthode de *fast primual-dual* (*fast-PD*), introduite par Nikos Komodakis et Georgios Tziritas en 2007 dans [125], repose sur le principe de dualité en théorie de programmation linéaire. Ainsi, en plus du graphe *primal* du problème de MRF d'origine, un graphe *dual* à celui-ci est construit (sans rapport avec les graphes duaux à un ensemble de sommets ou d'arêtes définis dans la sous-section 2.2.1). Trouver la solution optimale au problème se fait en cherchant la configuration solution pour les graphes qui minimise l'écart entre eux.

La recherche simultanée sur ces deux graphes permet d'obtenir une très bonne approximation du minimum global en un temps annoncé comme dix fois moindre qu'avec les techniques antérieures.

Voir également [126, 127].

6.4 Conclusion du chapitre

Dans ce chapitre, nous avons rappelé comment un problème d'étiquetage par MRF pouvait se ramener à un problème de minimisation d'énergie et comment celui-ci peut être résolu, selon les cas, par coupes minimales de graphes. Cette résolution ne donne cependant le minimum global que dans le cas d'un étiquetage binaire ou celui des multi-levels, autrement le résultat n'en est qu'une approximation bornée.

Il a également été rappelé que ce type de résolution de ne se limite pas au cas isotrope d'une métrique ou une semi-métrique mais pouvait ainsi résoudre une bien plus large classe de problèmes.

Quelques légères améliorations, comme mentionné dans les remarques 6.9 et 6.11, ont été apportées dans ce chapitre par rapport aux articles originaux de sorte, principalement, à diminuer au maximum le nombre de sommets et d'arcs de chaque graphe construit et ainsi accélérer la recherche d'une coupe minimale dessus.

Enfin, nous avons pu voir que malgré le développement de nouvelles techniques, la minimisation par coupes minimales a sans doute un avenir prometteur par sa simplicité de mise en place et sa rapidité d'exécution.

Renaissance d'image

7

La complétion d'image (*inpainting* en anglais, aussi appelée *retouching* ou *repairing*), introduite dans [25], consiste à compléter les régions manquantes, occultées ou endommagées d'une image à partir de ses régions intactes de sorte à obtenir en résultat une image qui soit visuellement plausible.

Les moyens d'arriver à un tel résultat sont, pour certains, étroitement liés à ceux utilisés pour la synthèse de textures qui peut se ramener à un problème de complétion d'images où seule une toute petite région texturée aurait été préservée. Le but est alors, à partir d'un échantillon de texture, de construire une image de dimensions quelconques ayant la même texture (et la même échelle) que l'échantillon.

De nombreuses techniques de complétion d'images essaient de propager le contenu des régions intactes bordant les régions à reconstruire à l'intérieur de celles-ci par interpolation ou extrapolation. Ces techniques ont souvent recours à des méthodes variationnelles, qui sont adaptées pour compléter des régions manquantes de petite taille et peu texturées.

Le second type de techniques employées en complétion d'images va chercher dans toute la région intacte de l'image un modèle à reproduire, ou directement une région à copier, dans la zone manquante. Ces techniques reposent souvent sur des méthodes statistiques, qui sont adaptées pour des régions manquantes avec des textures de type stochastique (voir le spectre de textures de la figure 7.1 pour une illustration de ce que sont les textures stochastiques).

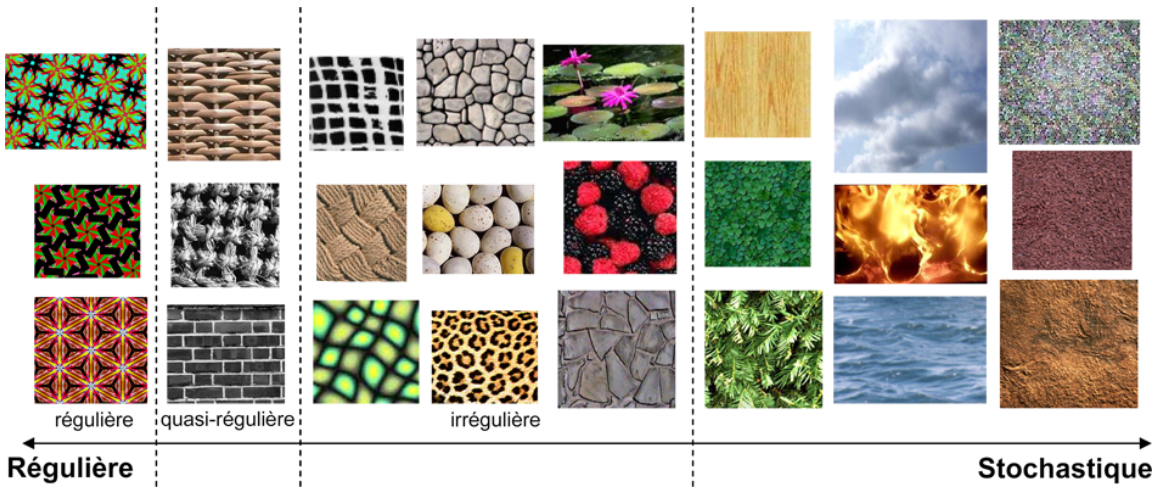


Figure 7.1 – Spectre de textures (illustration issue de [139]).

Les méthodes variationnelles sont grandement inspirées par le modèle élastique d'Euler [159], une formulation mathématique permettant de retrouver, à partir de points de

départ et d'arrivée ainsi que de leur normales, certains plans manquants grâce au principe de continuité [114].

Dans [142], l'approche utilisée, appelée *image disocclusion*, vise à décomposer l'image en isophotes (i.e. des lignes de même intensité lumineuse) et à les compléter dans la partie occultée grâce au modèle élastique [159].

Ce concept fut exploré sous sa forme géométrique dans [25] par l'intermédiaire d'équations aux dérivées partielles propageant les informations de l'image le long des isophotes dans la partie à reconstruire.

Une méthode similaire fut présentée dans [47] où une approche par diffusion orientée par courbure permet de compléter la partie manquante de l'image.

D'autres méthodes apparentées furent proposées dans [43, 46].

Enfin, une variante du cadre de travail de Mumford-Shah fut proposé dans [81] pour compléter les structures manquantes.

Alors que ces méthodes sont plutôt populaires et offrent de bonnes performances pour compléter des régions étroites, leur principale limitation est de ne pouvoir gérer des régions manquantes plus larges ou encore la présence de textures autour de celles-ci.

Un tour d'horizon des méthodes de complétion d'image par méthodes variationnelles présentées ci-dessus peut être trouvé dans [24].

Les méthodes statistiques sont issues de la synthèse de texture, un problème en quelque sorte relié à la complétion d'image puisqu'il peut être considéré comme compléter une image dont seule une petite région texturée aurait été préservée. Le but est alors, à partir d'un échantillon de texture, de construire une image de dimensions quelconques ayant la même texture que l'échantillon.

L'idée principale est de reproduire le motif de l'échantillon à une échelle locale fixée comme exploité dans [208].

Cette reproduction peut être réalisée à l'aide d'opérateurs basés sur des filtres comme dans [104, 215].

Ce problème fut, ces dernières années, souvent formulé en termes de champs aléatoires de Markov comme dans [80].

Enfin, certaines approches consistent à coller ensemble, côte à côte, des copies de morceaux de l'échantillon d'entrée. Ceci fut réalisé avec une optimisation par coupes minimales de graphes dans [79, 130].

En considérant que les régions manquantes peuvent être reconstruites à partir de régions de l'image intacte à la même échelle, cette dernière méthode est directement adaptable pour la complétion d'image comme cela a été fait dans [61, 106]. Une variante de ce modèle permettant de définir manuellement les structures principales à reconstruire fut présentée dans [195].

Ainsi, à l'opposé des méthodes variationnelles, les méthodes statistiques permettent d'obtenir de bien meilleurs résultats quand des textures entrent en jeu ou qu'il y a de larges régions à reconstruire.

Notre approche, baptisée *renaissance d'image*, vise à combiner les approches variationnelles et statistiques par le biais d'un problème de minimisation que l'on résout par α-*expansions* (voir section 6.2.2.2). Comme pour les dernières méthodes statistiques présentées, nous considérons que les parties manquantes de l'image peuvent être complétées par "copier-coller" de petites régions de l'image en provenance de ses parties intactes. Le côté variationnel se retrouve dans le fait que les régions utilisées comme patches pour remplir la zone à reconstruire sont déterminées par un calcul de similarité entre leur voisinage et celui de la région à compléter, qui est intacte et sert de "point de départ" à la complétion. Ceci a pour conséquence d'assurer la bonne continuité entre les parties intactes et celles à reconstruire de l'image étant donné qu'une variation similaire des couleurs existe ailleurs

dans l'image. Bien entendu, ce principe de continuité est également appliqué entre les différents patches qui remplissent la région à reconstruire, mais avec une priorité décroissante au fur et à mesure que l'on s'éloigne des régions intactes de l'image, remplaçant ainsi, de façon implicite, la notion de propagation progressive liée aux méthodes variationnelles.

Pour ce faire, nous avons reformulé le problème de complétion d'images sous la forme d'un problème d'étiquetage où chaque pixel de la zone à reconstruire se voit attribuer une étiquette correspondant à une translation dans l'image indiquant le pixel dont la valeur doit être copiée, comme illustré dans la figure 7.2. Nous avons alors formulé une énergie à minimiser permettant de résoudre notre problème tel qu'exposé ci-dessus (voir section 7.1). Néanmoins, de par le très grand nombre de combinaisons possibles, la résolution directe de ce problème serait des plus coûteuses en temps. C'est pourquoi, afin de réduire le nombre de combinaisons possibles avant de procéder aux *α-expansions* (voir section 7.2.2), nous procédons à une présélection des zones, dans l'image intacte, les plus susceptibles d'être choisies pour être copiées dans la région à reconstruire. Celle-ci est réalisée grâce à une technique de filtre à particules permettant l'évolution et le suivi de multiples hypothèses évaluées par une fonction de densité probabiliste (voir section 7.2.1).

Notre méthode a donc l'avantage de gérer correctement les textures ainsi que les grandes régions à compléter tout en assurant une bonne continuité des couleurs entre les régions intactes et les régions à reconstruire comme illustré dans la section 7.3.

Le travail présenté dans ce chapitre, fait en collaboration avec Nikos Paragios, a fait l'objet d'un rapport de recherche [9], d'une publication [10] ainsi que d'une présentation orale à la 18$^{\text{ème}}$ *International Conference on Pattern Recognition* qui s'est tenue à Hong-Kong (Chine) en Août 2006.

FIGURE 7.2 – Copie de régions dans la partie à reconstruire.

Sommaire du chapitre

Figures du chapitre

7.1 Complétion d'image en tant que problème de minimisation

7.1.1 Problème d'étiquetage lié à la complétion d'image

Soit une image à compléter $\mathcal{I}$ et un masque associé $\mathcal{M}$ représentant le sous-ensemble de pixels $p_{\mathcal{M}} \subset \mathcal{D}(\mathcal{I})$ de $\mathcal{I}$ nécessitant d'être reconstruits partiellement. Le masque $\overline{\mathcal{M}}$ représente le sous-ensemble de pixels $\overline{p_{\mathcal{M}}}$ de $\mathcal{I}$ qui sont intacts, seront donc conservés et qui serviront de sources d'échantillons pour la reconstruction des régions de $\mathcal{M}$.

Comme expliqué précédemment, nous considérons que les parties manquantes de l'image peuvent être reconstituées par copies de petites régions en provenance des parties intactes.

La renaissance d'image pour $\mathcal{I}$ sur $\mathcal{M}$ consiste donc à créer une image $\mathcal{I}'$ telle que :

$$\mathcal{I}'(p) = \begin{cases} \mathcal{I}(p') \text{ avec } p' \in \overline{p_{\mathcal{M}}} & \text{si } p \in p_{\mathcal{M}}, \\ \mathcal{I}(p) & \text{sinon.} \end{cases} \tag{7.1}$$

Cette reconstruction peut se traduire sous forme d'un problème d'étiquetage.

Notons $t_1, \ldots, t_n$ les n translations de l'image I permettant à au moins un de ses pixels translatés de se retrouver dans la région à reconstruire indiquée par le masque $\mathcal{M}$.

Notre problème d'étiquetage peut être formulé avec :

- l'ensemble de sites $\mathcal{P} = p_{\mathcal{M}} = \{p_1, \ldots, p_m\}$ étant l'ensemble des pixels des régions à reconstruire, et
- l'ensemble d'étiquettes $\mathcal{L} = \{t_1, \ldots, t_n\}$ étant l'ensemble des translations possibles permettant de copier un pixel de la partie intacte dans la partie à reconstruire.

Pour chaque pixel $p \in \mathcal{P}$ de coordonnées u_p dans $\mathcal{I}$, l'ensemble d'étiquettes $\mathcal{L}_p$ représente l'ensemble des translations permettant d'attribuer une valeur au pixel p par copie de celle du pixel p' de coordonnées $u_{p'} = u_p - t_i$ avec $t_i \in \mathcal{L}_p$.

Les translations de l'ensemble d'étiquettes permettent donc de déterminer, pour chaque pixel de la zone à reconstruire, les pixels de la zone intacte qui sont des candidats potentiels pour la copie de leur valeur.

Le voisinage considéré dans notre problème d'étiquetage peut être n'importe quel voisinage lié à l'image mais, dans notre cas, nous considérerons le 4-voisinage.

On note $x = \{x_1, \ldots, x_m\} \in \{t_1, \ldots, t_n\}^m$ une combinaison possible de notre problème d'étiquetage et $\mathcal{X}$ l'ensemble des combinaisons possibles. Par "possible", on entend que la combinaison remplit les conditions de translations mentionnées ci-avant.

7.1.2 Energie liée à la complétion d'image

Comme expliqué dans la section 6.1.1, nous définissons une énergie à minimiser mesurant l'optimalité d'une combinaison d'étiquetage selon les critères de continuité de couleurs énoncés plus haut. Celle-ci sera donc liée au principe de continuité :peut se décomposer en deux parties :

- entre les régions à copier dans la zone à recontruire et la partie intacte de l'image (voir sous-section 7.1.2.2), et
- entre les différentes régions à copier dans la zone à reconstruire (voir sous-section 7.1.2.3).

7.1.2.1 Mesure de dissimilarité entre régions d'images

Avant de définir cette énergie nous présentons différentes façons, de plus en plus complexes, de mesurer la dissimilarité entre deux pixels de la région à reconstruire selon

les translations que l'on souhaite leur attribuer.

Soient p et q deux pixels voisins de la zone à reconstruire (i.e. $\{p,q\} \in \mathcal{V}$ et $p,q \in p_{\mathcal{M}}$) de coordonnées respectives u_p et u_q.
Soient t_i et t_j deux translations de $\mathcal{I}$ valables pour p et q (i.e. $t_i, t_j \in \mathcal{L}_p \cap \mathcal{L}_q$).

Il est à souligner que, par définition, nous désirons que cette mesure de dissimilarité soit nulle si nous avons $t_i = t_j$.

Une mesure simple de dissimilarité, proposée dans [130], consiste à faire la somme des normes entre les valeurs des pixels p et q translatés :

$$\delta''_{\{p,q\}}(t_i,t_j) = \|\mathcal{I}(u_p - t_i) - \mathcal{I}(u_p - t_j)\| + \|\mathcal{I}(u_q - t_i) - \mathcal{I}(u_q - t_j)\| \tag{7.2}$$

Une première amélioration de cette mesure consiste à étendre cette mesure au-delà des deux pixels concernés uniquement. On ne considère plus uniquement les pixels p et q mais un petit nombre de pixels supplémentaires pris de part et d'autres dans leur prolongement et pondéré par une gaussienne normalisée centrée entre p et q.
Ainsi, dans le cas du 4-voisinage, cela revient à prendre en compte des pixels supplémentaires sur la même ligne ou la même colonne que p et q.
Pour déterminer les coordonnées des pixels à prendre en compte, on rajoute une nouvelle translation t entre la translation nulle t_0 et la translation maximale T pour les pixels pondérés par la gaussienne les plus éloignés de p et q. En considérant que le pixel p est de coordonnées inférieures à celles du pixel q, on obtient alors la mesure suivante :

$$\delta'_{\{p,q\}}(t_i,t_j) = \sum_{t=t_0}^{T} \frac{1}{\sigma\sqrt{2\pi}} \exp\Big(-\frac{|t|^2}{2\sigma^2}\Big)\Big(\|\mathcal{I}(u_p - t_i - t) - \mathcal{I}(u_p - t_j - t)\| + \|\mathcal{I}(u_q - t_i + t) - \mathcal{I}(u_q - t_j + t)\|\Big) \tag{7.3}$$

Une autre amélioration, présentée dans [130], est l'ajout d'un terme inversement proportionnel aux normes des gradients de l'image sur les positions des pixels p et q translatés. En se basant sur la dernière mesure de dissimilarité proposée on obtient alors :

$$\delta_{\{p,q\}}(t_i,t_j) = \frac{\delta'_{\{p,q\}}(t_i,t_j)}{|\nabla\mathcal{I}(p - t_i)| + |\nabla\mathcal{I}(p - t_j)| + |\nabla\mathcal{I}(q - t_i)| + |\nabla\mathcal{I}(q - t_j)| + 1} \tag{7.4}$$

Cet ajout a pour conséquence d'obtenir une mesure de dissimilarité plus faible là où il y a un gradient fort pour les translatés respectifs des pixels p et q. Ceci représente le fait qu'un changement de régions dans l'étiquetage est moins visible s'il se trouve sur des contours de l'image (qui ont donc un fort gradient).

Par la suite, nous utiliserons cette dernière mesure, $\delta_{\{p,q\}}$, pour quantifier la dissimilarité entre les différentes régions copiées dans la zone à reconstruire de l'image.
Nous considérerons comme norme entre deux valeurs de pixels la distance euclidienne dans l'espace RVB.

Remarque 7.1.
Il est à noter que, selon les critères énoncés ci-avant et en considérant une translation t_k de $\mathcal{I}$ valable également pour p et q (i.e. $t_k \in \mathcal{L}_p \cap \mathcal{L}_q$), nous avons :

$$\delta_{\{p,q\}}(t_i,t_i) + \delta_{\{p,q\}}(t_j,t_k) \leq \delta_{\{p,q\}}(t_i,t_k) + \delta_{\{p,q\}}(t_j,t_i) \tag{7.5}$$

La fonction $\delta_{\{p,q\}}$ est donc régulière.
Cette propriété reste valable pour toute métrique sur l'espace des couleurs au lieu de la distance euclidienne que nous utilisons.

Dans nos applications, le calcul de dissimilarité se fait sur un total de 10 pixels, soit 4 de plus de part et d'autre de p et q sur la même ligne ou colonne et nous prenons la valeur $\sigma = 2$.

7.1.2.2 Terme de continuité avec la zone intacte de l'image

Dans le but de propager l'information contenue par les pixels de $p_{\mathcal{M}}$ dans ceux de $\overline{p_{\mathcal{M}}}$ en maximisant la continuité des couleurs, les régions utilisées pour reconstruire le pourtour de la zone manquante de l'image doivent avoir été copiées à partir d'une zone ayant un contenu similaire à celui bordant la zone qu'elles vont compléter. Il est donc nécessaire de minimiser la différence entre leurs voisinages respectifs.

Soit $p_{\mathcal{B}}$ l'ensemble de pixels de $p_{\mathcal{M}}$ qui sont voisins de $\overline{p_{\mathcal{M}}}$.

En prenant une combinaison $x \in \mathcal{X}$, notre mesure de continuité entre région reconstruite et région intacte de l'image s'écrit comme la somme sur les pixels de la zone à compléter qui suit :

$$E_{pourtour}(x) = \sum_{p \in p_{\mathcal{M}}} \rho_p(x_p) \tag{7.6}$$

avec, pour $p \in p_{\mathcal{M}}$ et $t \in \mathcal{L}_p$:

$$\rho_p(t) = \begin{cases} \sum\limits_{q \in \{\mathcal{V}_p \cap \overline{p_{\mathcal{M}}}\}} \delta_{\{p,q\}}(x_p, t_0) & \text{si } p \in p_{\mathcal{B}}, \\ 0 & \text{sinon.} \end{cases} \tag{7.7}$$

avec t_0 la translation nulle puisque, par définition, p' est dans la partie intacte de l'image (i.e. $p' \in \overline{p_{\mathcal{M}}}$).

7.1.2.3 Terme de continuité entre les régions copiées

La sous-section 7.1.2.2 présente la mesure de la continuité entre les pixels copiés dans la zone à reconstruire et ceux de la zone intacte de l'image. Nous allons maintenant présenter la mesure de la continuité entre les pixels copiés dans la zone à reconstruire.

Comme annoncé précédemment, afin que notre reconstruction simule une propagation du contenu de l'image intacte dans la région à reconstruire, nous souhaitons que l'étiquetage des pixels au bord de la zone à compléter soit prépondérant sur ceux situés plus au centre de cette même zone.
Pour ce faire, nous pondérons notre mesure de dissimilarité présentée en sous-section 7.1.2.1 de façon inversement proportionnelle à la distance euclidienne du couple de pixels voisins considérés au bord de la zone à reconstruire.

En prenant une combinaison $x \in \mathcal{X}$, notre mesure de continuité dans la région à reconstruire s'écrit comme la somme sur les arêtes adjacentes à deux facettes du maillage qui suit :

$$E_{interieur}(x) = \sum_{\substack{p,q \in p_{\mathcal{M}} \\ \{p,q\} \in \mathcal{V}}} \frac{\delta_{\{p,q\}}(x_p, x_q)}{\mathcal{D}(p, \overline{\mathcal{M}}) + \mathcal{D}(q, \overline{\mathcal{M}})} \tag{7.8}$$

où $\mathcal{D}(p, \overline{\mathcal{M}})$ est la distance euclidienne entre le pixelé p et le masque $\overline{\mathcal{M}}$.

Comme $\delta_{\{p,q\}}(x_p, x_q) = 0$ si $x_p = x_q$, minimiser ce terme favorise donc un étiquetage similaire sur des pixels voisins.

7.1.2.4 Minimisation globale

Nous déduisons des sous-sections 7.1.2.2 et 7.1.2.3 l'énergie globale à minimiser qui suit :

$$\begin{aligned} E(x) &= E_{pourtour}(x) + \gamma\, E_{interieur}(x) \\ &= \sum_{p \in p_{\mathcal{M}}} \rho_p(x_p) + \gamma \sum_{\substack{p,q \in p_{\mathcal{M}} \\ \{p,q\} \in \mathcal{V}}} \frac{\delta_{\{p,q\}}(x_p, x_q)}{\mathcal{D}(p, \overline{\mathcal{M}}) + \mathcal{D}(q, \overline{\mathcal{M}})} \end{aligned} \tag{7.9}$$

où γ est une constante de pondération entre les deux termes (typiquement $\gamma = 1$).

La fonction $E_{pourtour}$ est le terme direct de l'énergie alors que la fonction $E_{interieur}$ en est le terme de voisinage (voir section 6.1.2).

Le résultat obtenu par cette minimisation est une partition des pixels de la zone à reconstruire qui favorise le continuité des couleurs entre la partie intacte de l'image et la région à reconstruire tout en rendant aussi discrète que possible la jointure présente entre les pixels d'étiquettes différentes.

7.2 Mise en application

Le nombre de translations possibles, et donc d'étiquettes, s'avère souvent très élevé. C'est pourquoi, afin d'éviter une étape de minimisation trop coûteuse, nous effectuons une pré-selection des étiquettes les plus probables d'être utilisées.

Cette pré-sélection se fait par le biais de filtres à particules, comme expliqué dans la section 7.2.1, avant d'utiliser une technique de minimisation par coupes minimales de graphe, comme expliqué dans la section 7.2.2.

7.2.1 Sélection des régions candidates

Nous appelons *région candidate* une région de la partie intacte de l'image ayant une bonne similarité de contenu avec une région située à la frontière avec la partie de l'image à reconstruire. Une telle région correspond à une étiquette (i.e. la translation de la frontière à la région candidate) qui pourra être utilisée pour compléter l'image avec une bonne continuité des couleurs à la frontière entre la partie intacte et la partie à reconstruire de l'image.

Pour procéder à la sélection de régions candidates, nous utilisons un filtre à particules, qui est une technique séquentielle de Monte-Carlo introduite dans [98]. Cette technique statistique itérative repose sur des fonctions de densité de probabilité estimées par les états des itérations antérieures d'un ensemble d'échantillons (voir également [13, 73, 74, 75]).

7.2.1.1 Principe du filtre à particules

Le principe d'un filtre à particules repose sur la répétition de deux étapes à opérer sur un ensemble d'échantillons en nombre fixé (les particules) :

1. Pondération : pondérer les échantillons d'après une fonction de densité de probabilité liée au problème, et
2. Rééchantillonnage : générer un nouvel ensemble d'échantillons (en nombre égal) par le biais d'une perturbation des caractéristiques des précédents et avec les propriétés suivantes :

– plus le poids d'un échantillon est élevé et plus il est susceptible de générer de nouveaux échantillons,
– la perturbation appliquée à un nouvel échantillon est inversement proportionnelle au poids de l'échantillon qui la génère ainsi qu'au nombre d'itérations déjà effectuées.

Une illustration de ces deux étapes se trouve dans la figure 7.3.

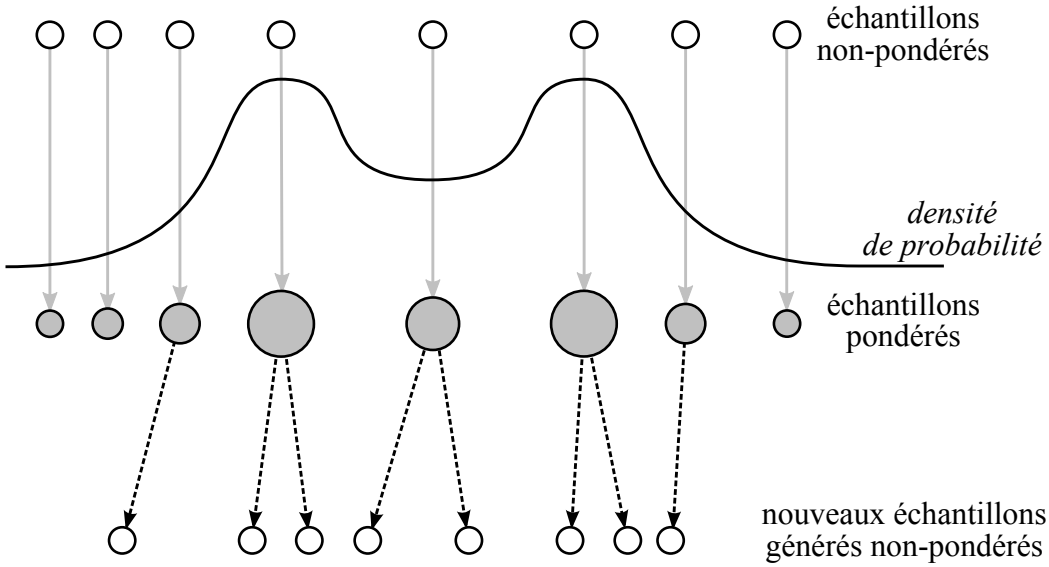


FIGURE 7.3 – Principe d'une itération de filtre à particules.

Ainsi, chaque échantillon permet de déterminer une probabilité et l'on considère que si celle-ci est élevée alors une solution du problème se trouve à proximité, d'où la réduction des perturbations. De même, la diminution de ces dernières au fur et à mesure des itérations permet une convergence autour des estimations les plus probables.

Plusieurs critères d'arrêts peuvent être appliqués à un filtre à particule : nombre d'itérations prédéfini, seuillage sur la pondération des échantillons, invariance entre deux itérations,...

7.2.1.2 Filtre à particules appliqué à la sélection de régions candidates

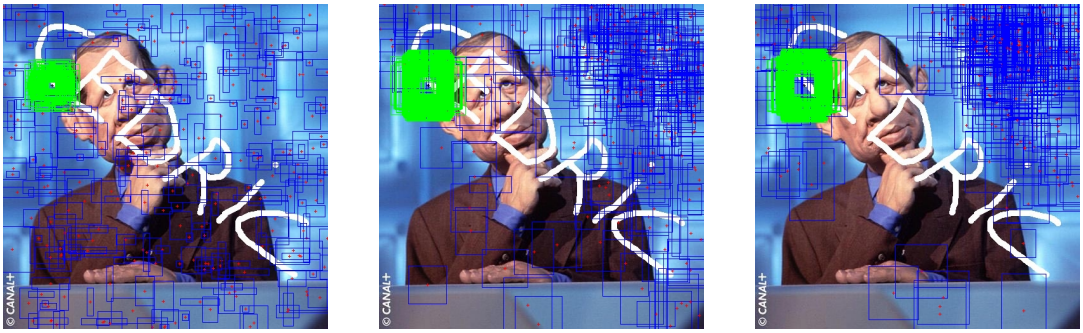
Dans le but de rechercher des zones de la partie intacte de l'image ayant un contenu similaire à celui jouxtant la partie à compléter, nous considérons un ensemble de pixels pris à la frontière entre les deux. Pour chacun de ces pixels, nous utilisons un filtre à particule afin d'obtenir un nombre donné de régions candidates ayant les meilleures similitudes avec la région située autour du pixel concerné.

Pour chaque pixel $p \in p_{\mathcal{B}}$, chaque particule k qui lui est liée a pour paramètre :
– une translation t telle que $t \in \mathcal{L}_p$ qui fait donc référence à un pixel q de coordonnées $u_q = u_p - t$,
– une largeur ℓ, et
– une hauteur h.

La pondération de la particule correspond à l'évaluation de la similarité, dans la partie intacte de l'image, entre la région située autour de p et celle autour de q. Celle-ci est calculée par l'inverse de la somme des carrés des différences entre les valeurs des pixels des rectangles de taille $l \times h$ centrés, respectivement, en p et en q.

Lors de la phase de génération des nouveaux échantillons, les particules avec la meilleure similitude à la région du pixel de référence seront les plus probables à servir d'origine au nouvel échantillon auquel on appliquera une perturbation. Celle-ci est une modification bornée de la translation t et des dimensions ℓ et h du rectangle de mesure de la similarité. Comme exposé précédemment, cette perturbation est d'autant plus faible que le poids de la particule est grand et que le nombre d'itérations est élevé. Ainsi, une particule avec un poids élevé, et donc une bonne similitude avec la région du pixel référence, aura tendance à générer de nouveaux échantillons avec une translation très proche de celle d'origine. Ceci a pour effet de faire converger, au fur et à mesure des itérations, les particules dans les zones de la partie intacte de l'image ayant les meilleures similarités avec les alentours du pixel de référence.

Une illustration de l'évolution d'un filtre à particule appliqué tel que défini ci-dessus se trouve dans la figure 7.4.



(a) (b) (c)

FIGURE 7.4 – Visualisation d'une centaine de particules pour la recherche de zone similaires au bord de la zone à reconstruire (cadres verts) dans le restant de l'image (cadres bleus). Au fur et à mesure, les zones de recherche convergent dans des régions similaires.

Après un nombre fixé d'itérations au-delà duquel les particules ne peuvent plus évoluer, nous ne conservons dans $\mathcal{L}_p$ que les translations des particules ayant les plus fortes pondérations.

7.2.2 Minimisation par *α-expansions*

Maintenant que nous avons un ensemble d'étiquettes réduit, nous pouvons procéder à la recherche d'une reconstruction optimale en cherchant par minimisation de l'énergie de l'équation 7.9 la combinaison solution du problème d'étiquetage offrant la meilleure complétion d'image.

La méthode de minimisation que nous utilisons est celle des *α-expansions*. Ceci est possible d'après le corollaire 6.13 et la remarque 7.1 qui met en évidence que le terme de voisinage $\delta_{\{p,q\}}$ de l'énergie $E(x)$ est régulier et que cette dernière peut donc être minimisée par *α-expansions*. La minimisation se déroule comme expliqué dans la section 6.2.2.2) à l'exception que nous ordonnons les étiquettes traitées en commençant par celle dont la similarité au bord de la zone à reconstruire est la plus forte (i.e. celle dont la particule correspondante était de poids le plus fort). Ceci a pour conséquence de mettre en place très rapidement les étiquettes correspondant à de bonnes conditions de continuité avec la partie intacte de l'image et ainsi d'accélérer le calcul global puisqu'il n'est alors plus consacré, principalement, qu'au calcul des meilleures transitions entre

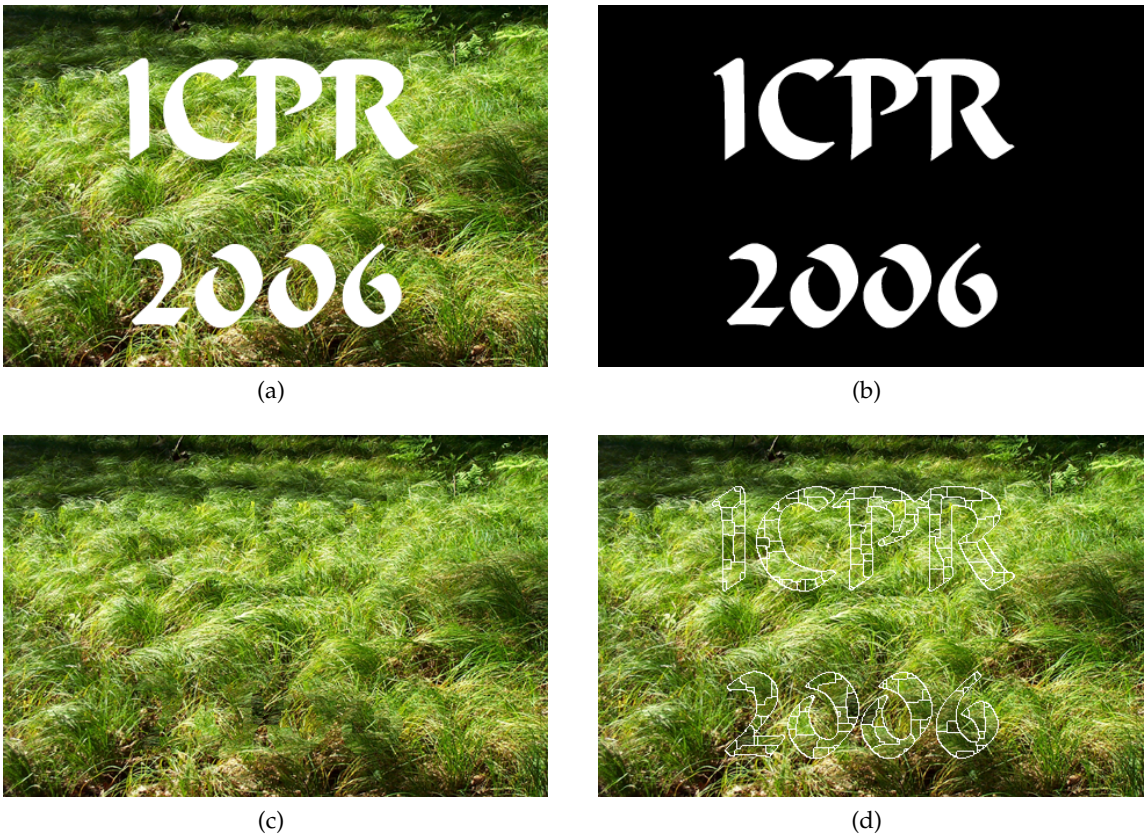
régions d'étiquettes différentes au sein de la zone à compléter.

Une fois la partition calculée, le rendu de l'image reconstruite se fait simplement en copiant, pour chaque pixel de la zone à compléter, celui de la partie intacte pointé par la translation correspondant à l'étiquette attribuée.

Cet affichage peut très bien être fait à chaque itération d'*α-expansion* si l'on souhaite suivre l'évolution de la reconstruction.

7.3 Résultats

Nous exposons dans cette section les résultats de complétion d'images obtenus avec notre méthode. Certaines images, présentées dans les figures 7.7, 7.9, 7.11, 7.13 et 7.15, sont des exemples classiques en complétion d'image. Il est ainsi possible de comparer nos résultats à ceux obtenus dans [25], présentés, respectivement, dans les figures 7.8, 7.10, 7.12, 7.14 et 7.16. Nous présentons également nos résultats sur de nouveaux exemples, tels que ceux des figures 7.5 et 7.6.



(a) (b)

(c) (d)

Figure 7.5 – Renaissance d'image d'une photographie d'herbes avec : (a) l'image à compléter ; (b) le masque de la zone à compléter ; (c) l'image reconstruite ; (d) la partition de la zone reconstruite.

Ces résultats nous permettent de nous rendre compte que notre méthode de complétion d'images gère très bien les zones texturées (voir, notamment, les figures 7.5 et 7.15, cette dernière montrant l'exemple cité comme limite à la méthode variationnelle proposée dans [25] - voir la reconstruction "floue" sur la figure 7.16) y compris lorsque la région à

(a) (b)

(c) (d)

FIGURE 7.6 – Renaissance d'image d'une photographie de marionnette avec : (a) l'image à compléter ; (b) le masque de la zone à compléter ; (c) l'image reconstruite ; (d) la partition de la zone reconstruite.

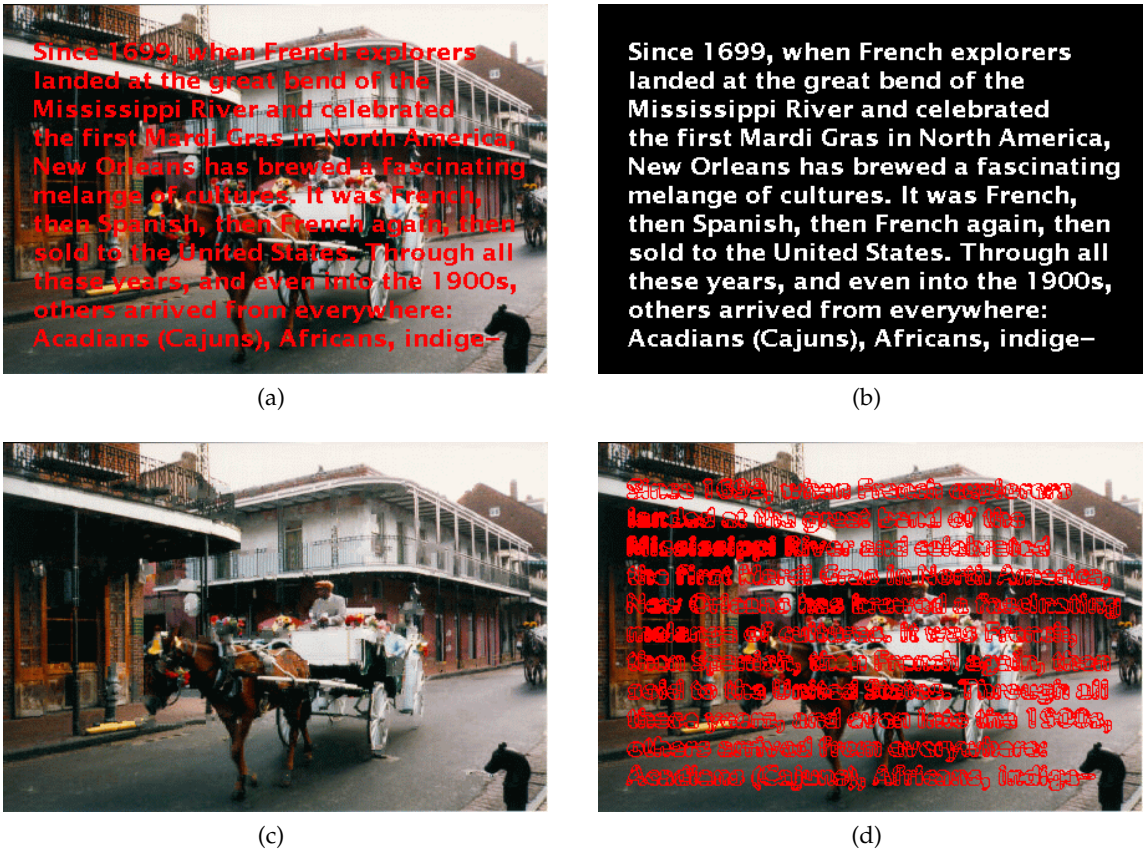
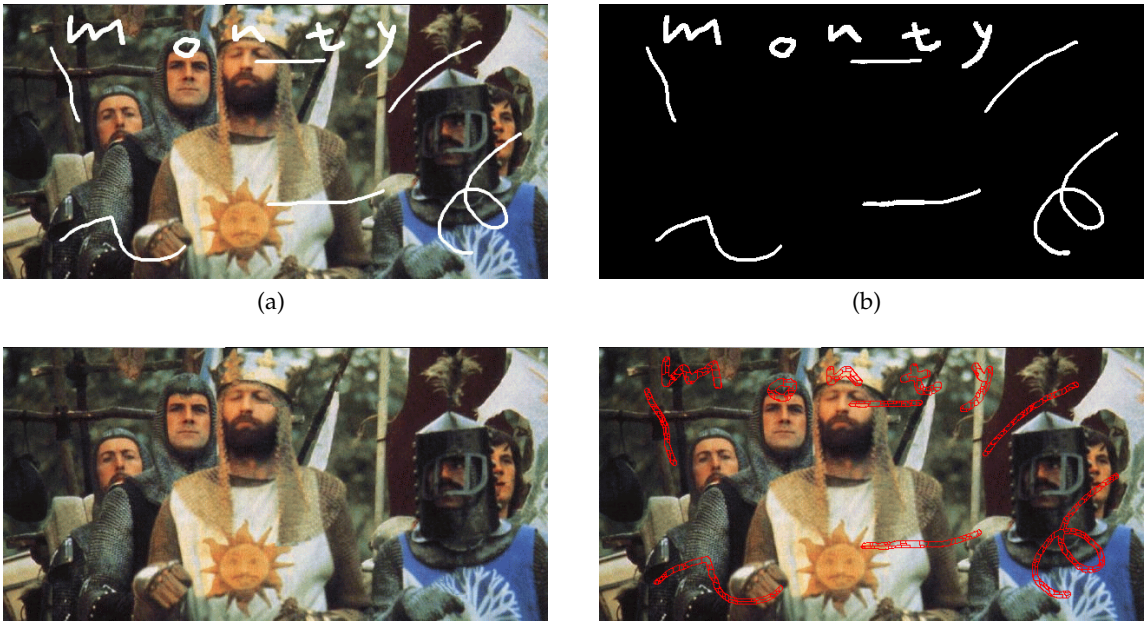


FIGURE 7.7 – Renaissance d'image d'une photographie de charrette avec : (a) l'image à compléter ; (b) le masque de la zone à compléter ; (c) l'image reconstruite ; (d) la partition de la zone reconstruite.

FIGURE 7.8 – Complétion d'image d'une photographie de charrette avec la méthode variationnelle proposée dans [25] à comparer à notre résultat de la figure 7.7c.

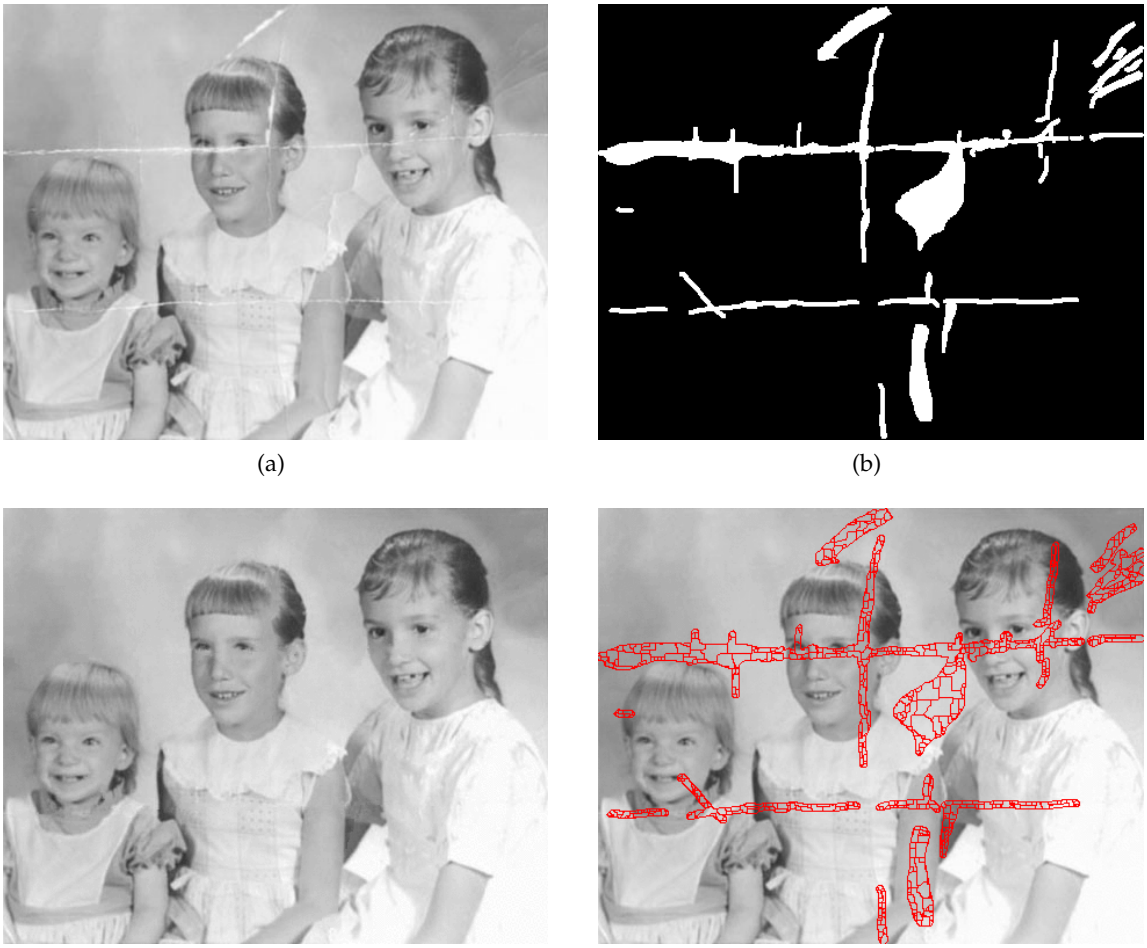


(a) (b)

(c) (d)

FIGURE 7.9 – Renaissance d'image d'une photographie tirée du film *Sacré Graal !* des Monty Python avec : (a) l'image à compléter ; (b) le masque de la zone à compléter ; (c) l'image reconstruite ; (d) la partition de la zone reconstruite.

FIGURE 7.10 – Complétion d'image d'une photographie tirée du film *Sacré Graal !* des Monty Python avec la méthode variationnelle proposée dans [25] à comparer à notre résultat de la figure 7.9c.



(a) (b)

(c) (d)

FIGURE 7.11 – Renaissance d'image d'une photographie de trois soeurs avec : (a) l'image à compléter ; (b) le masque de la zone à compléter ; (c) l'image reconstruite ; (d) la partition de la zone reconstruite.

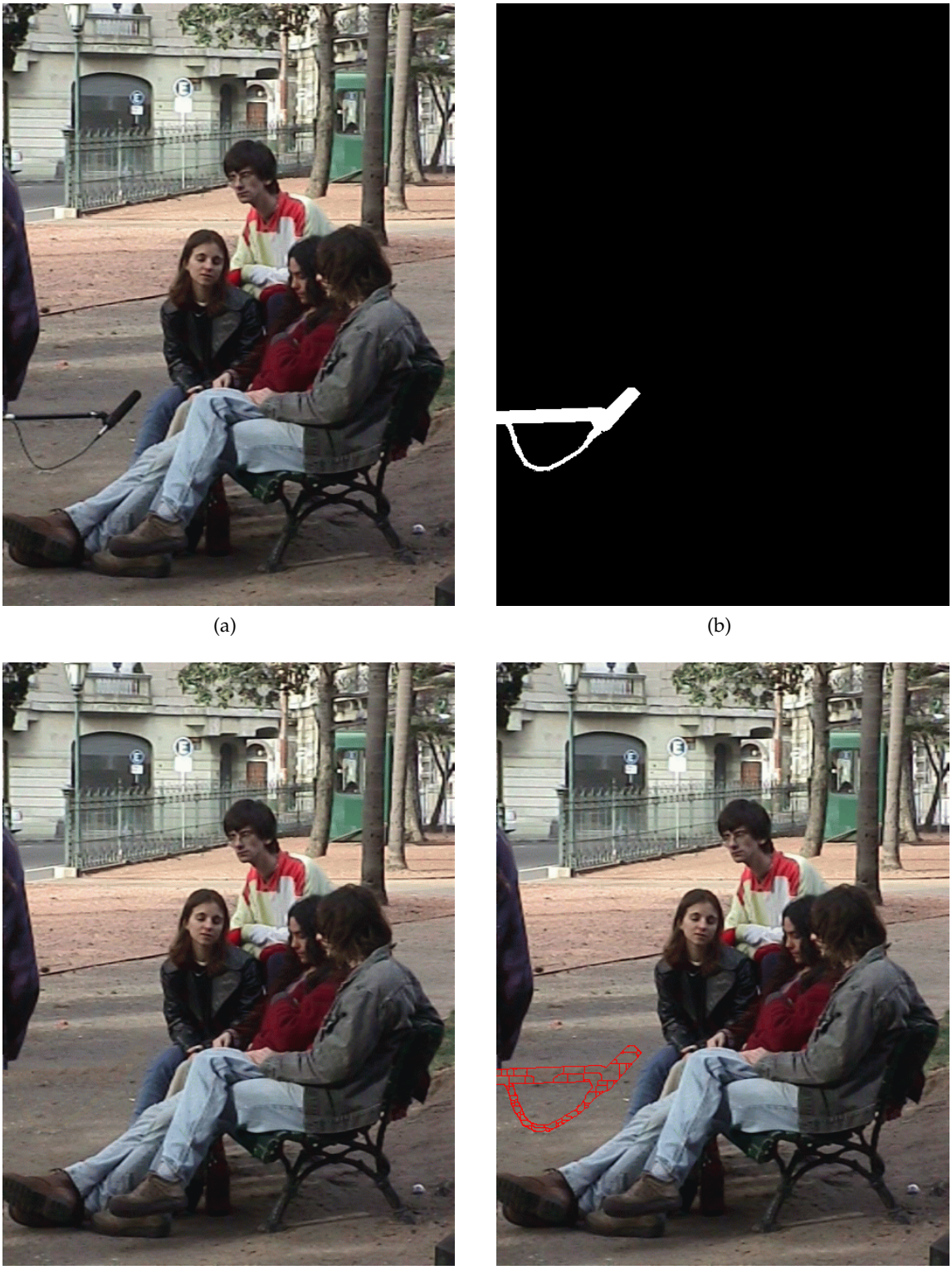
FIGURE 7.12 – Complétion d'image d'une photographie de trois soeurs avec la méthode variationnelle proposée dans [25] à comparer à notre résultat de la figure 7.11c.

(a) (b)

(c) (d)

FIGURE 7.13 – Renaissance d'image d'une illustration de manga avec : (a) l'image à compléter ; (b) le masque de la zone à compléter ; (c) l'image reconstruite ; (d) la partition de la zone reconstruite.

FIGURE 7.14 – Complétion d'image d'une illustration de manga avec la méthode variationnelle proposée dans [25] à comparer à notre résultat de la figure 7.13c.



(a) (b)

(c) (d)

FIGURE 7.15 – Renaissance d'image d'une photographie d'interview avec : (a) l'image à compléter ; (b) le masque de la zone à compléter ; (c) l'image reconstruite ; (d) la partition de la zone reconstruite.

FIGURE 7.16 – Complétion d'image d'une photographie d'interview avec la méthode variationnelle proposée dans [25] à comparer à notre résultat de la figure 7.15c.

reconstruire est plus large que dans les exemples classiques (voir figure 7.5).

Des animations du déroulement de notre méthode et de l'évolution de la reconstruction sur ces exemples sont disponibles à cette adresse : http://certis.enpc.fr/~allene/research-inpainting.html.

Le temps d'exécution de notre méthode est de l'ordre de quelques minutes. Celui-ci dépend de la taille de l'image, de celle de la zone à reconstruire, du nombre de recherches de zones similaires et du nombre de particules utilisées pour chacune d'elles. De plus, le temps de calcul est réparti de manière quasi-égale entre la phase de recherche des régions candidates par filtre à particules et celle de minimisation avec les étiquettes sélectionnées. Il est à noter que si l'on supprime la phase de pré-sélection, le temps de calcul de la minimisation sur l'ensemble complet d'étiquettes dépasse la dizaine d'heures.

De par la définition du problème, il ne nous a pas été possible de définir une mesure de qualité du résultat. En effet, on cherche à obtenir une reconstruction qui soit *visuellement plausible*. Prendre une image, la dégrader, faire la reconstruction pour pouvoir ensuite quantifier la différence entre l'image originale et l'image reconstruite (avec, par exemple, la somme des carrés des différences) nous a été suggéré par un relecteur lors de la soumission de notre article. Cependant en imaginant une zone de couleur uniforme sur l'image d'origine qui se retrouve dans la zone à compléter. Il est tout à fait possible d'imaginer une reconstruction offrant une couleur légèrement différente mais uniforme sur cette même zone donner une mesure de qualité moins bonne mesure de qualité qu'une reconstruction offrant des discontinuités de couleur mais dont la moyenne retombe exactement sur la couleur de la zone d'origine. Ainsi, ce type de mesure ne semble pas approprié puisque visuellement, l'ajout de discontinuités serait plus choquant pour l'oeil.

De plus, déterminer une quantification correcte du problème de complétion d'images

reviendrait à trouver une nouvelle méthode pour la réaliser puisqu'il suffirait alors d'utiliser cette quantification comme énergie à minimiser pour obtenir l'image reconstruite "parfaite".

7.4 Conclusion

La méthode de complétion d'images que nous avons proposée permet de reconstruire une partie d'une image, indiquée par un masque, si l'on suppose que le contenu manquant ou détérioré peut être retrouvé dans la partie intacte de l'image. Cette méthode a l'avantage de gérer correctement la complétion de zones texturées puisque celles-ci sont copiées à partir de zones de l'image ayant la même texture.

Une méthode reposant sur le même principe de complétion d'images consistant à copier des morceaux de l'image intacte dans la partie à reconstruire par le biais d'une minimisation imitant le procédé de propagation fut proposé en 2006 par Nikos Komodakis et Georgios Tziritas dans [124]. La différence majeure avec notre méthode est que celle-ci utilisait une version modifiée de la *loopy belief propagation* (voir section 6.3.4) comme moyen de minimisation.

Depuis sa création, la complétion d'images s'est étendue au champ de la reconstitution de vidéos [120, 162]. Le principe de base demeure le même, à savoir reconstruire une partie manquante, occultée ou détériorée d'une séquence vidéo de façon visuellement plausible. Néanmoins, dans ce cas, il faut tenir compte de contraintes temporelles de sorte à ne pas avoir de variations incohérentes entre deux images successives dans la vidéo.

La méthode que nous proposons ici pourrait très bien être adaptée à ce cadre. Cependant, une application directe rendrait la résolution extrêmement coûteuse en temps de calcul étant donné le grand nombre d'étiquettes possibles puisqu'on ne se contente plus de chercher les pixels à copier dans la même image mais dans l'ensemble des images formant la séquence vidéo. Une première approche basant cette complétion sur la segmentation de couches de mouvements (voir, par exemple, [76, 77]) fut proposée dans [214]. Cette décomposition permettrait ainsi de limiter grandement les zones de recherche des pixels à copier dans la zone à reconstituer, rendant alors sans doute le calcul de minimisation réalisable en des temps acceptables.

Textures pour maillages 3D de reconstruction multi-vues

8

La reconstruction 3D à partir de vues multiples d'un objet ou d'une scène est un problème très étudié en vision par ordinateur qui consiste à produire une représentation tridimensionnelle de cet objet ou de cette scène par l'extraction de la géométrie et des attributs visuels de plusieurs photographies prises sous des angles différents.

Notre cerveau est capable de percevoir les distances par triangulation à partir de nos yeux. C'est également sur ce principe que repose la reconstruction 3D d'un objet ou d'une scène. En effet, si l'on connaît les positions à partir desquelles ont été prises les photos ainsi que les paramètres des appareils utilisés, les coordonnées d'un point qui apparaît dans ces photos pourront alors être déterminées.

Une illustration d'objet 3D perçu à travers différentes prises de vues se trouve dans la figure 8.1.

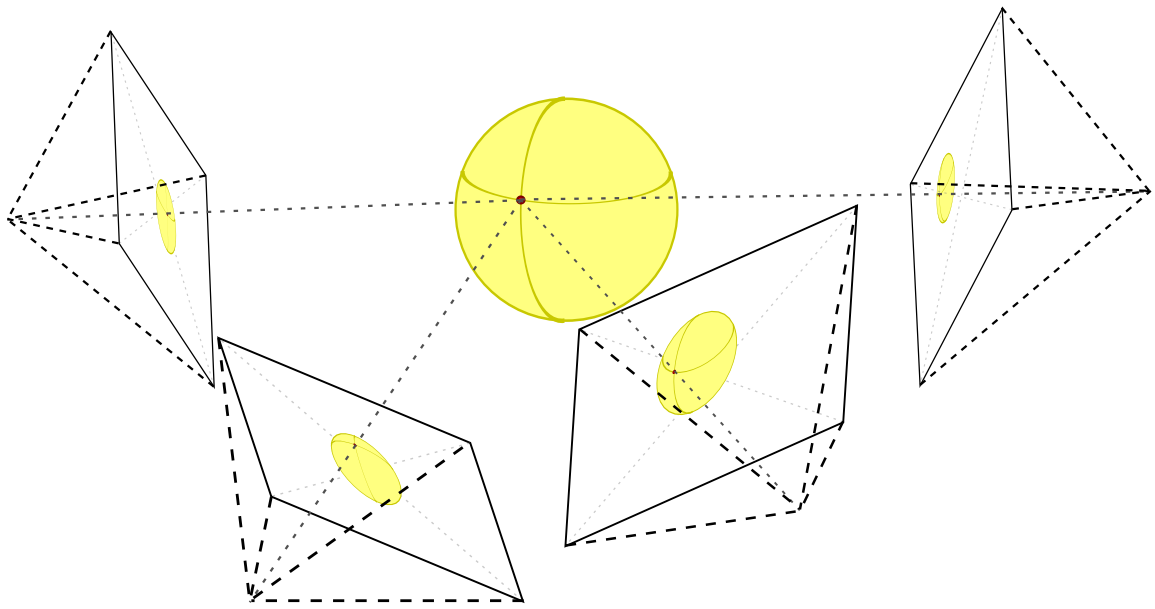


Figure 8.1 – Principe de reconstruction 3D.

Les paramètres intrinsèques des appareils servant à prendre les différents vues (tels que, par exemple, focale, distorsion,...) et leurs positions exactes durant les prises de vues, aussi appelées paramètres extrinsèques, ne sont pas nécessairement connus mais peuvent être retrouvés. Pour cela, on procède à une phase de calibration sur mire pour les paramètres intrinsèques et à une autocalibration pour les paramètres extrinsèques. Cette

dernière peut se faire par appariement de points d'intérêts. Cela consiste à mettre en correspondance entre les images des points facilement reconnaissables d'une image à l'autre (comme des intersections de droites par exemple) s'ils y apparaissent.

Une illustration d'appariements de points entre deux images d'une même scène se trouve dans la figure 8.2.

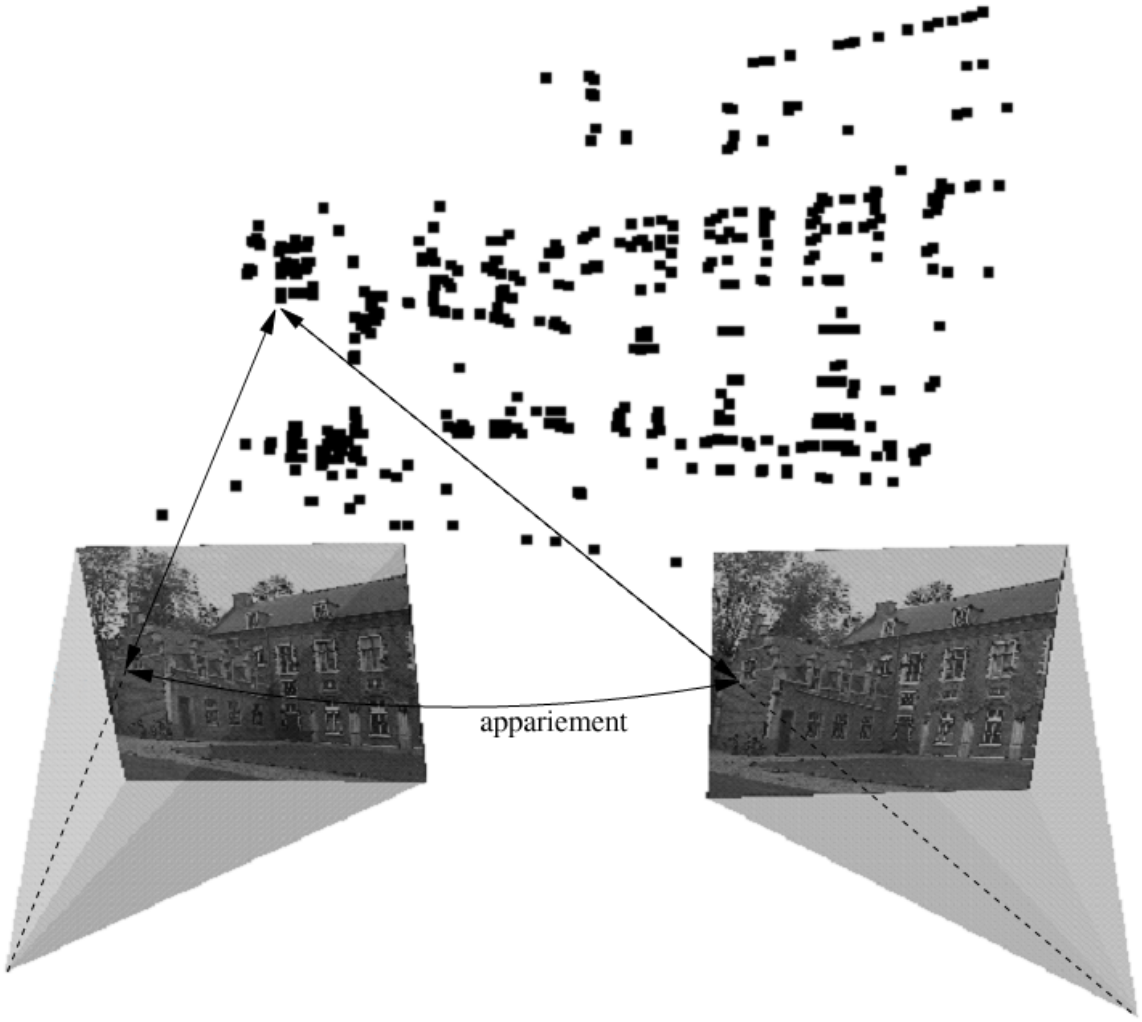


FIGURE 8.2 – Appariement de points d'intérêts pour l'autocalibration. Illustration issue de [167].

Une fois les appariements de points entre images effectués, la résolution d'un système d'équation permet de retrouver dans l'espace 3D le nuage de points d'intérêts et la position des différentes prises de vues.

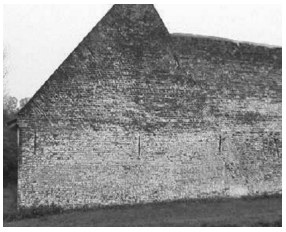
Une illustration représentant le nuage de points d'intérêts et les positions des prises de vues correspondantes se trouve dans la figure 8.3.

Finalement, en retrouvant tous les paramètres des différentes prises de vues, il est alors possible de créer un maillage, composé de nombreux triangles également appelés facettes, représentant l'objet ou la scène et d'y projeter une texture en provenance de parties des différentes images utilisées pour la reconstruction.

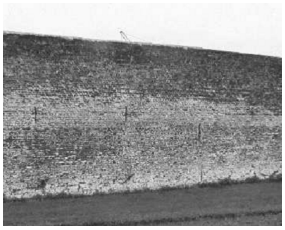
Une illustration représentant la reconstruction 3D et la texture obtenue à partir de l'exemple autocalibré dans la figure 8.3 se trouve dans la figure 8.4.

(a)

(b)



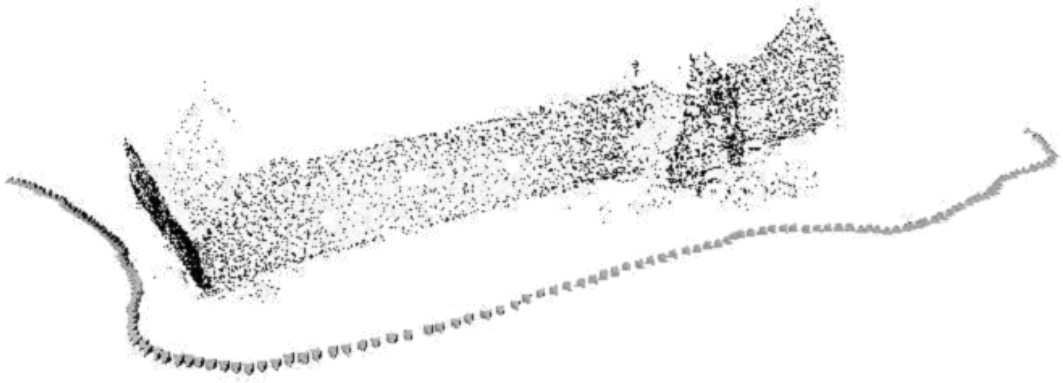
(c)



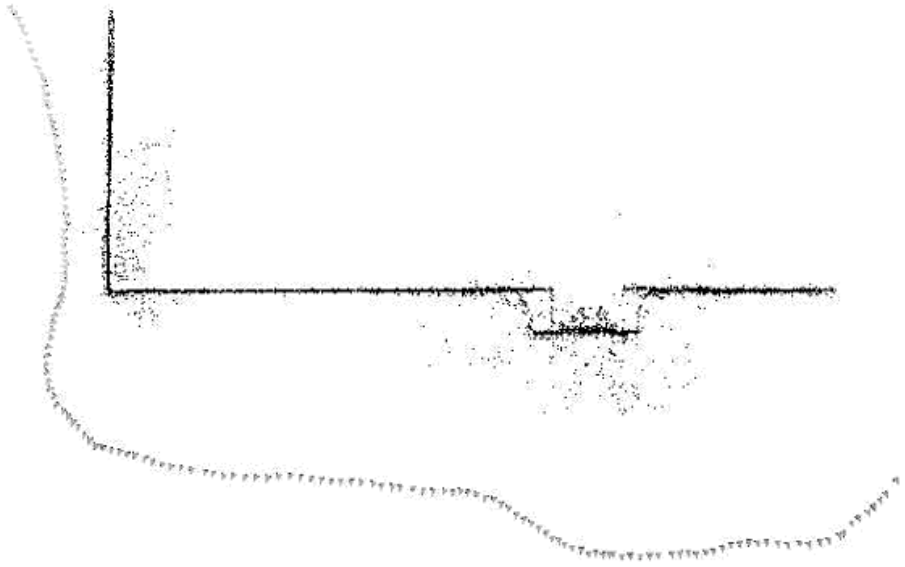
(d)

(e)

(f)

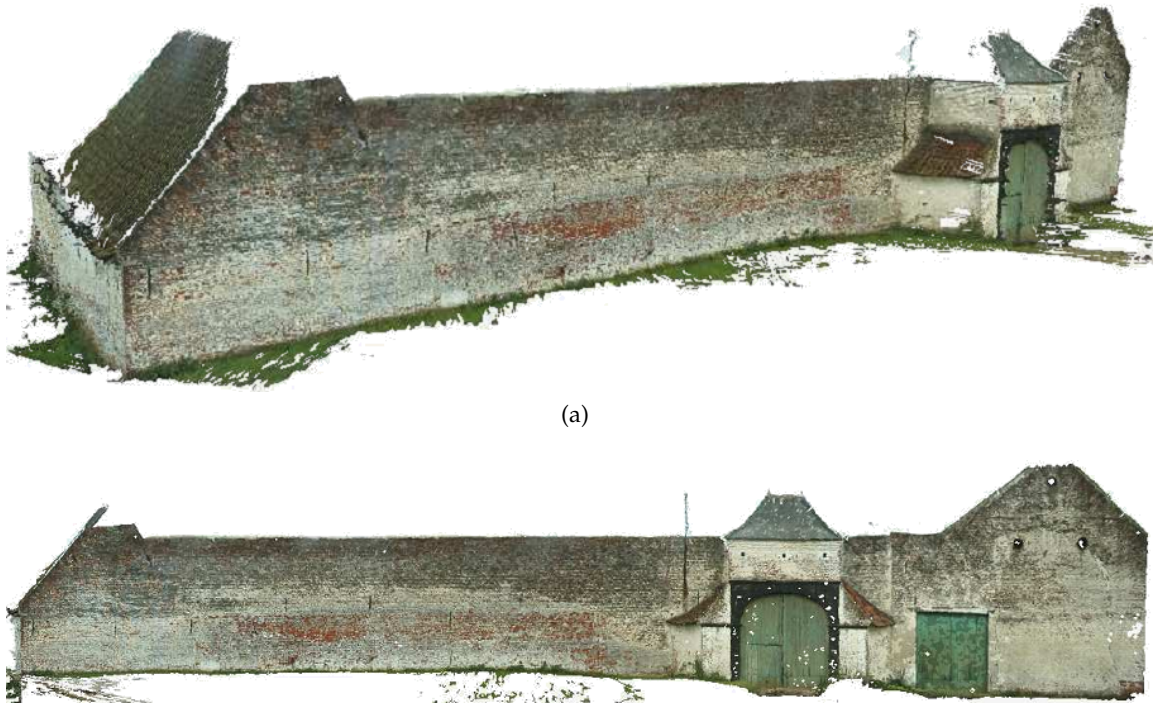


(g)



(h)

FIGURE 8.3 – (a) à (f) Six photographies d'un corps de ferme parmi 150. (g,h) Deux vues du nuage de points d'intérêts ayant servi à l'autocalibration des 150 prises de vues représentées dans l'espace 3D par la rangée de pyramides. Illustrations issues de [168].



(a)

(b)

FIGURE 8.4 – Deux vues du modèle 3D obtenu par reconstruction multi-vues à partir des photographies et de l'autocalibration de la figure 8.3. Illustrations issues de [168].

Pour que la reconstruction 3D de l'objet ou de la scène soit complète, il faut appliquer une texture sur la maillage généré. C'est sur ce point particulier que se portent nos travaux présentés dans ce chapitre.

La capture, le rendu et l'affichage de la couleur d'un modèle 3D requiert, typiquement, de projeter sa surface sur un domaine bidimensionnel. Généralement, il est impossible de trouver une telle paramétrisation globale avec une distorsion acceptable, c'est pourquoi une structure d'atlas est souvent adoptée. Cela consiste en une partition de la surface en différentes régions connexes et d'une paramétrisation 2D par morceaux (voir, par exemple, [172] et les références qu'elle contient).

L'instanciation de ce problème à partir de vues multiples a fait l'objet de nombreuses attentions dans les communautés de vision par ordinateur et de synthèse d'image.

D'un côté, le fait que la reconstruction 3D repose sur de multiples images facilite le problème de paramétrisation : les transformations projectives de la surface dans les images d'entrée constituent des applications naturelles et optimales (voir [134, 173, 204]). Elles évitent le rééchantillonnage d'image et la perte de détails visuels, contrairement à d'autres approches basées sur d'autres paramétrisations (voir [19, 23, 109]).

D'un autre côté, dans le contexte de modélisation à base d'images, les discontinuités de couleurs aux frontières des régions sont un problème crucial dû aux imprécisions photométriques et géométriques (variations dans les conditions d'éclairage ou les paramètres de prises de vues, réflexion non-Lambertienne, calibration imparfaite des appareils, approximations du maillage,...). Calculer une moyenne pondérée des images dans les régions où plusieurs images différentes se superposent (voir [23, 68, 136, 166, 173]) n'est pas suffisant. En effet, à part si le modèle 3D est obtenu par un moyen d'une grande précision (avec, par exemple, des mesures de distance par laser) et que la calibration est très précise, cela entraîne des effets indésirables de flou ou des artefacts de type fantôme

(voir [23, 110, 136]).

Deux axes principaux de recherche ont été explorés dans le but de réduire la visibilité de ces frontières.

La première approche consiste en une optimisation de la détermination des régions. Certains travaux forcent les frontières entre régions à se trouver en des zones du maillage de forte courbure (voir [137, 190]). D'autres utilisent un terme de fidélité d'image [134, 213] de façon à chercher explicitement une partition de la surface qui induit le moins de discontinuités de couleur possible. La formulation de ce problème en tant qu'optimisation d'un champ aléatoire de Markov est des plus intéressantes puisqu'elle permet l'utilisation de méthodes de résolution qui ont fait l'objet de nombreux travaux. Cependant, ces travaux souffrent de l'absence de calcul au niveau des pixels des images : ils sont incapables de déterminer une continuité parfaite des couleurs.

La seconde voie d'amélioration s'effectue justement au niveau des pixels en procédant à une correction des couleurs pour ceux se trouvant aux alentours des frontières de régions. Un travail important dans cette catégorie, présenté par Adam Baumberg en 2002 dans [19], est une tentative d'extension du mélange d'images 2D [42] (voir sections précédentes) au cas des textures 3D. Cependant, ce travail passe à côté de l'importance de la détermination de manière optimisée de la partition et donc des régions. De plus, il ne définit pas de zones de transition de largeurs distinctes appropriées pour les différentes bandes de fréquences se restreignant par conséquent à la décompositionen en deux bandes de fréquences seulement pour limiter les artéfacts visuels de type fantôme.

Nous proposons une nouvelle méthode permettant de créer un atlas de texture de haute qualité sans jointure visible à partir d'un modèle 3D et d'un ensemble d'images calibrées. Cette méthode améliore celle utilisant les champs aléatoires de Markov présentée dans [134] et l'étend en ajoutant le mélange d'images multi-fréquences appliqué au cas de textures de modèles 3D. Par conséquent, nous obtenons un placement quasi-optimal des jointures qui se voient dotées d'une continuité des couleurs de part et d'autre. En voici les étapes principales :

1. Nous créons une partition des triangles du maillage en assignant à chacun d'eux une image dont la texture y sera projetée de sorte à minimiser la visibilité de la jointure susceptible d'apparaître quand deux triangles voisins se voient attribuer des images différentes (voir section 8.1).
2. Ensuite, nous procédons à un mélange d'images sur la texture obtenue de sorte à faire disparaître la jointure pouvant résulter de la première étape puisque celle-ci, bien que minimale, n'est pas nécessairement nulle. Pour se faire, il nous a fallu étendre le mélange multi-fréquences de deux images présenté dans [42] au cas d'un nombre quelconque d'images (voir section 8.2).
3. Enfin, dans la but de réduire l'espace mémoire occupé, nous construisons un atlas de la texture obtenue à l'étape précédente (voir section 8.3).

Nous montrons l'efficacité de notre méthode sur deux jeux de données considérés difficiles constitués de photographies de hautes résolutions de scènes réelles :

- l'"Aiguille du Midi" : 37 images (1000×1332) du fameux pic rocheux du Mont-Blanc situé Chamonix (France), copyright Bernard Vallet (www.bvallet.com).
- le "Château d'Ettlingen" : 19 images (1536×1024) du château d'Ettlingen (Allemagne), copyright Christoph Strecha, EPFL (http://cvlab.epfl.ch/~strecha/multiview/).

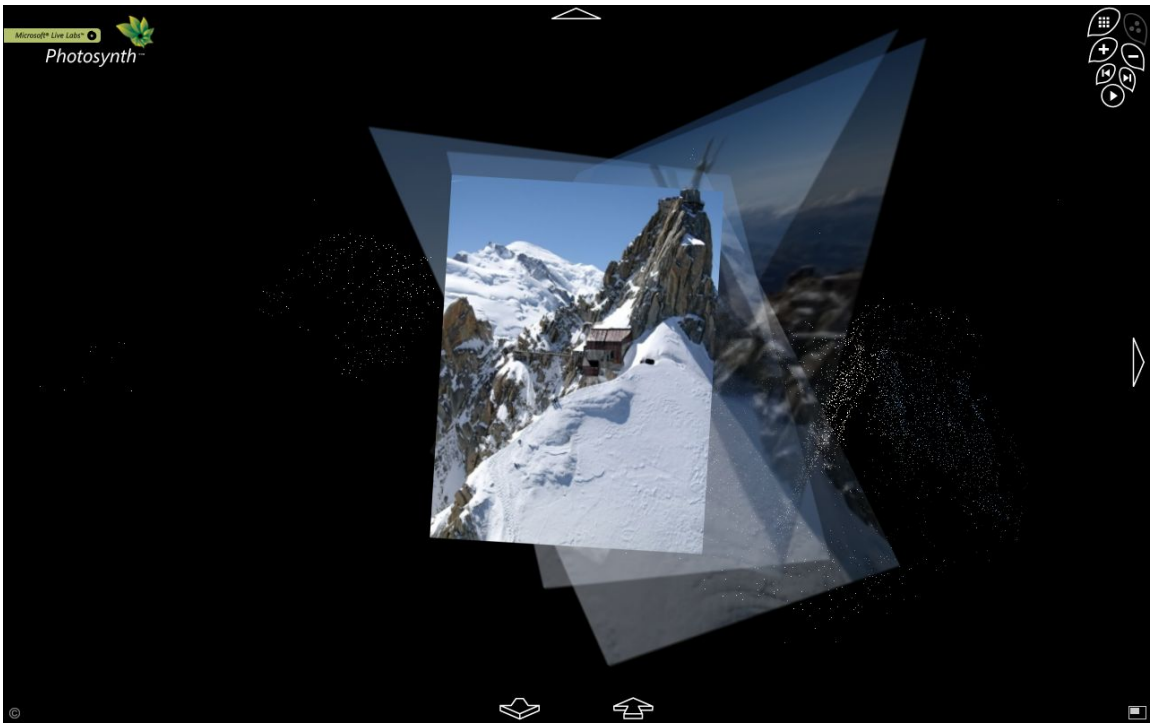
Une illustration montrant la projection d'images autocalibrées dans le nuage 3D de points d'intérêts pour ces deux jeux de données se trouve dans la figure 8.5 (autocalibration et affichage effectués avec *Photosynth* - `http://photosynth.net/`).

Pour obtenir des modèles géométriques précis de ces larges scènes de manière automatique, nous avons tout d'abord employé la méthode mise au point par Patrick Labatut, Jean-Philippe Pons et Renaud Keriven présentée dans [131] en 2007, qui est une technique de stéréovision multi-vues basée sur une recherche de points d'intérêts, une triangulation 3D de Delauney et une optimisation globale par coupe minimales de graphes. Les modèles obtenus étaient ensuite raffinés grâce à un maillage déformable par descente de gradient sur des scores de correspondance multi-vues selon la méthode de Jean-Philippe Pons, Renaud Keriven et Olivier Faugeras décrite dans [169] en 2007.
Une vue de ces maillages se trouve dans la figure 8.6.

Le travail présenté dans ce chapitre, fait en collaboration avec Jean-Philippe Pons et Renaud Keriven, a été publié [11] et fait l'objet d'une présentation orale à la 19ème *International Conference on Pattern Recognition* qui s'est tenue à Tampa (Floride) en Décembre 2008.

Sommaire du chapitre

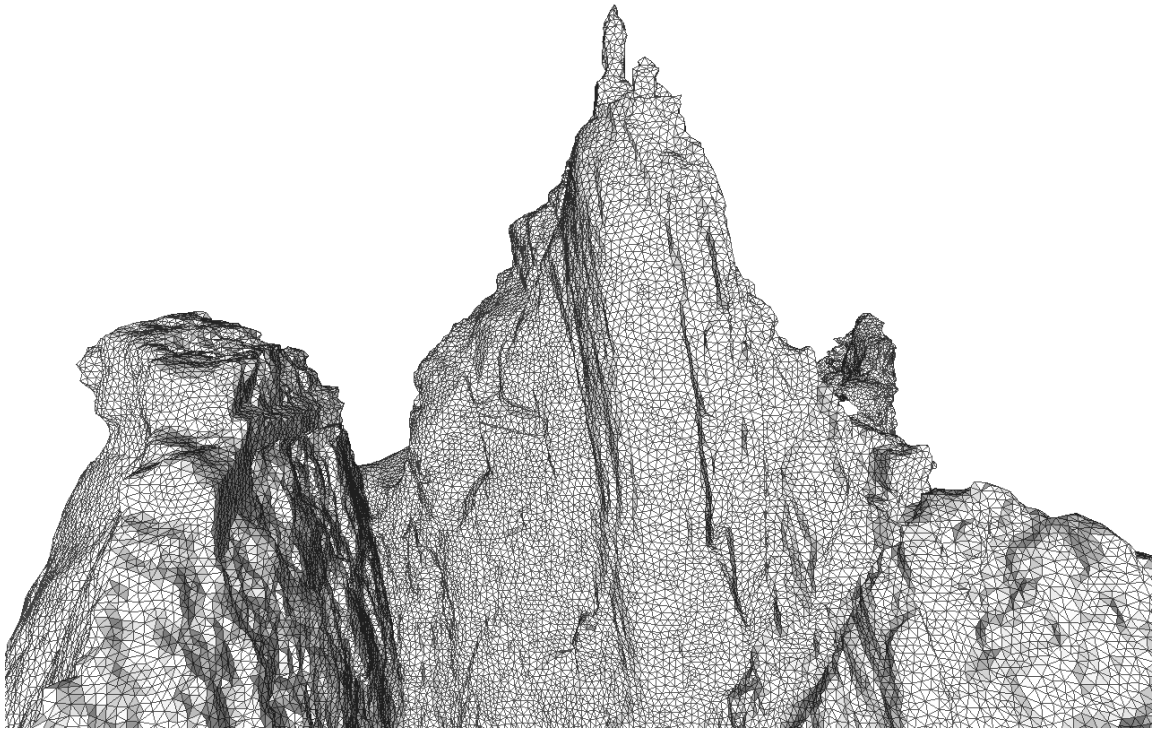
Figures du chapitre



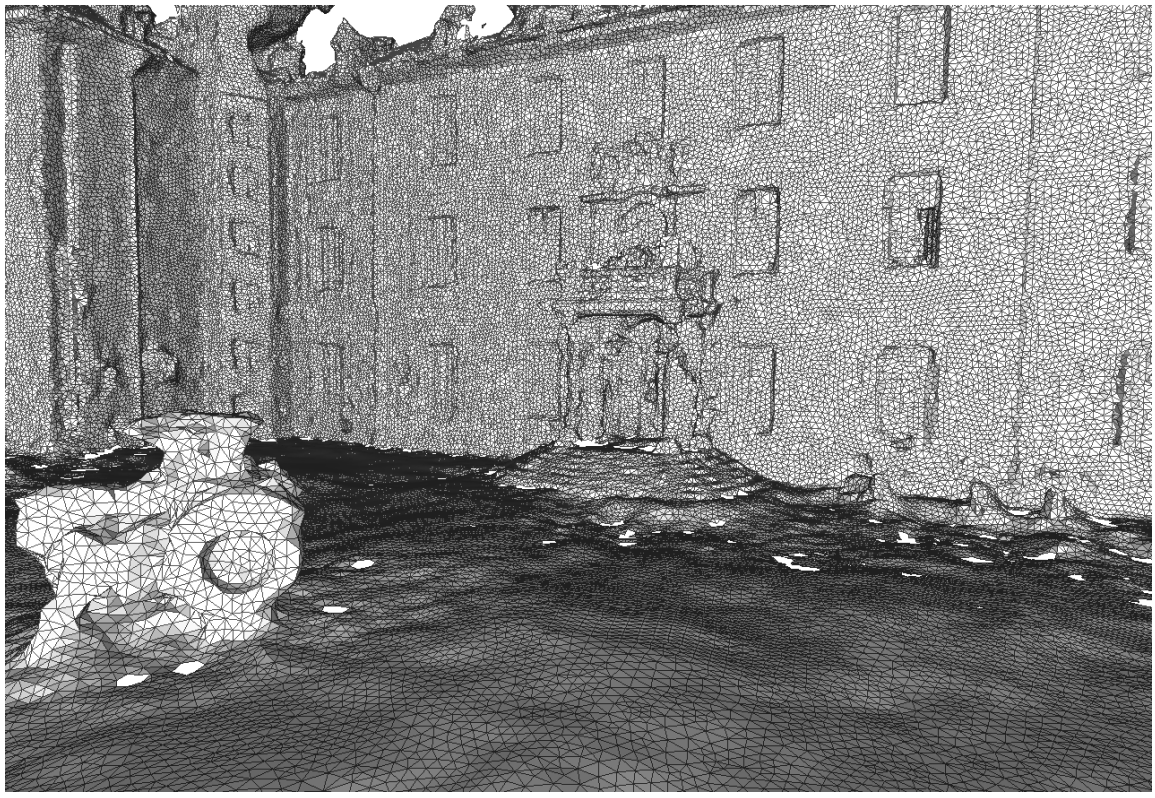
(a)

(b)

FIGURE 8.5 – Projection d'images autocalibrées dans le nuage de points d'intérêts utilisé pour : (a) l'Aiguille du Midi ; (b) le château d'Ettlingen. Autocalibration et images issues de Photosynth (http://photosynth.net/).



(a)



(b)

FIGURE 8.6 – Maillages de reconstruction 3D de : (a) l'Aiguille du Midi ; (b) le château d'Ettlingen.

8.1 Optimisation de la partition

La première étape de notre méthode s'approche de la méthode présentée dans [134] mais avec quelques améliorations et clarifications. On calcule une partition de la surface du maillage qui offre un bon compromis entre la qualité des détails visuels de la texture projetée sur chaque facette et la continuité de couleur aux frontières des régions. Cette optimisation est réalisée par *α-expansions*.

En pratique, cela consiste à assigner à chaque facette du maillage une des vues d'entrée dans laquelle elle est visible. Théoriquement, toute facette devrait apparaître dans au moins une image d'entrée. Ceci peut se traduire sous forme d'un problème d'étiquetage.

Notons $I_1, \ldots, I_n$ les images d'entrées calibrées et Π_i la projection de l'espace 3D dans l'image I_i. Nous supposons que la surface de l'objet ou de la scène est représentée par un maillage polygonal $\mathcal{M}$ avec les facettes $\mathcal{F} = \{f_1, \ldots, f_m\}$.
Notre problème d'étiquetage peut être formulé avec :
- l'ensemble de sites $\mathcal{P} = \mathcal{F} = \{f_1, \ldots, f_m\}$ étant l'ensemble des facettes du maillage, et
- l'ensemble d'étiquettes $\mathcal{L} = \{1, \ldots, n\}$ représentant l'ensemble des images $\{I_1, \ldots, I_n\}$ à disposition.

On note $x = \{x_1, \ldots, x_m\} \in \{1, \ldots, n\}^m$ une combinaison possible de notre problème d'étiquetage et $\mathcal{X}$ l'ensemble des combinaisons possibles. Par "possible", on entend que la combinaison remplit les conditions de visibilité mentionnées ci-avant.

Comme expliqué dans la section 6.1.1, nous définissons une énergie à minimiser mesurant l'optimalité d'une combinaison d'étiquetage selon les critères énoncés plus haut.

8.1.1 Maximisation des détails de la texture

Le terme direct de l'énergie est une mesure de la qualité des détails visuels de chaque facette.
Plutôt qu'une combinaison classique liée à la résolution de l'image, la distance de la prise de vue et l'angle entre la direction de la prise de vue et la normale de la facette, telle que, par exemple, celle proposée dans [134], nous adoptons une mesure à la fois plus aisée à interpréter et plus simple à calculer : le nombre total de **texels** (i.e. les éléments constituant une texture, qui sont l'équivalent des pixels d'une image) sur chaque facette.
Il est à mettre en évidence que ce critère simplifié prend en compte ceux présentés ci-avant puisqu'ils ont pour conséquence directe d'augmenter le nombre de pixels projetés sur la texture.

En prenant une combinaison $x \in \mathcal{X}$, notre terme direct s'écrit comme la somme sur les facettes qui suit :

$$E_{details}(x) = -\sum_{j=1}^{m} aire\left[\Pi_{x_j}(f_j)\right] \tag{8.1}$$

où $aire\left[\Pi_{x_j}(f_j)\right]$ mesure le nombre de pixels de l'image I_j recouverts par la projection de la facette f_j sur I_j, autrement dit, le nombre de texels de la texture de la facette f_j obtenue à partir de l'image I_j.

8.1.2 Minimisation de la visibilité des jointures

Le terme de voisinage de l'énergie est, quant à lui, une mesure de la continuité de la couleur sur les arêtes du maillage qui appartiennent à deux facettes ayant des étiquettes différentes.

Notons $a_{\{j,k\}}$ une arête du maillage qui soit adjacente aux facettes f_j et f_k. Si ces deux facettes se voient assigner à des images différentes, autrement dit, si elles ont des étiquettes différentes (i.e. $x_j \neq x_k$), alors il est probable qu'une discontinuité de couleurs apparaisse sur cette arête. On dit alors que $a_{\{j,k\}}$ est une arête de jointure ou encore qu'elle est sur la jointure entre deux régions. Dans la but de minimiser la visibilité des jointures, le terme de voisinage est défini comme l'intégrale de l'écart entre les images le long de la jointure.

En prenant une combinaison $x \in \mathcal{X}$, notre terme de voisinage s'écrit comme la somme sur les arêtes adjacentes à deux facettes du maillage qui suit :

$$E_{jointure}(x) = \sum_{a_{\{j,k\}} \in \mathcal{A}(\mathcal{M})} g_{a_{\{j,k\}}}(x_j, x_k) \tag{8.2}$$

avec, pour toute arête $a \in A(\mathcal{M})$ adjacente à deux facettes et pour toutes les étiquettes $\ell, \ell' \in \mathcal{L}$ représentant des images visibles pour ces facettes :

$$g_a(\ell, \ell') = \int_a \| I_\ell(\Pi_\ell(v)) - I_{\ell'}(\Pi_{\ell'}(v)) \| \, dv \tag{8.3}$$

où la norme utilisée correspond à la distance euclidienne dans l'espace de couleurs RVB.

8.1.3 Minimisation globale

Nous déduisons des sections 8.1.1 et 8.1.2 l'énergie globale à minimiser qui suit :

$$\begin{aligned} E(x) &= E_{details}(x) + \gamma \, E_{jointure}(x) \\ &= -\sum_{j=1}^{m} aire\left[\Pi_{x_j}(f_j)\right] + \gamma \sum_{a_{\{j,k\}} \in \mathcal{A}(\mathcal{M})} g_{a_{\{j,k\}}}(x_j, x_k) \end{aligned} \tag{8.4}$$

où γ est une constante de pondération entre les deux termes (typiquement $\gamma = 1$).

Pour toutes les étiquettes $\ell, \ell', \ell'' \in \mathcal{L}$ répondant aux conditions précédemment données pour une arête $a \in A(\mathcal{M})$ adjacente à deux facettes, nous avons :

$$g_a(\ell, \ell) + g_a(\ell', \ell'') \leq g_a(\ell, \ell'') + g_a(\ell', \ell) \tag{8.5}$$

En conséquence de quoi, d'après le corollaire 6.13, le terme de voisinage g_a de l'énergie $E(x)$ étant régulier, cette dernière peut donc être minimisée par *α-expansions*.

Il est à noter que cette propriété reste valable pour toute métrique sur l'espace des couleurs au lieu de la distance euclidienne que nous utilisons.

Cette propriété de régularité n'est pas précisée dans [134] alors que, en pratique, elle a des conséquences considérables puisqu'elle permet l'utilisation des *α-expansions* comme méthode de minimisation d'énergie (voir la sous-section 6.2.2.2).

Le résultat obtenu par cette minimisation est une partition des facettes du graphe qui rend la plus discrète possible la jointure présente sur la texture du maillage. Néanmoins, bien qu'aussi discrète que possible, la jointure n'en est pas pour autant invisible,

d'où la nécessité d'un traitement supplémentaire comme celui présenté dans la section 8.2.

Il est à préciser que, si l'on ne prend en compte que le terme direct $E_{details}$ dans notre énergie (i.e. on considère que $E_{jointure} = 0$), cela revient à projeter indépendamment sur chaque facette l'image qui lui offre la meilleure résolution de texture. Cette approche "naïve" servira de référence dans la section 8.4 présentant nos résultats (voir figures 8.20 et 8.21).

8.2 Mélange d'images par décomposition multi-fréquences

Le mélange d'images (*blending* en anglais) est souvent utilisé pour reconstituer un panoramique ou une fresque pris à partir de plusieurs photographies séparées (voir, par exemple, [39]) mais également dans le but de faire du montage ou du trucage photo (voir, par exemple, [178]).

Certaines méthodes se contentent ne modifient pas les images d'entrées mais n'en conservent qu'une partie en cherchant la délimitation entre les deux images à mélanger de sorte à ce que la jointure en résultant soit la plus discrète possible (comme, par exemple, dans [194] où la jointure optimale est déterminée par LPE aux endroits de fort gradient ou encore dans [99] où la recherche de la jointure optimale se fait par coupes minimales après une première délimitation faite par LPE). D'autres méthodes, au contraire, pour une délimitation donnée, modifient les images de part et d'autre de sorte à diminuer la visibilité de la jointure.

Etant donné que l'optimisation présentée dans la section 8.1 permet d'obtenir la jointure la moins visible possible sur la texture, cette deuxième étape sera consacrée à la faire disparaître en modifiant la texture de part et d'autre des frontières déterminées précédemment.

En 1983, Peter J. Burt et Edward H. Adelson présentèrent dans [42] une méthode de mélange de deux images basée sur la décomposition en différentes bandes de fréquences de l'image et à pratiquer un mélange différent pour chacune d'elles. Notre méthode est une généralisation de leur travail pour des images de taille quelconque et en nombre supérieur à deux. Nous l'avons ensuite appliqué à notre problème à partir des images d'entrée par le biais des fonctions de projection sur la maillage.

Nous commencerons la présentation de cette procédure en détaillant la décomposition d'une image en plusieurs bandes de fréquences telle que décrite dans [41]. Le but initial était alors de proposer une méthode de compression d'images par différenciation de bandes de fréquences utilisant des algorithmes de codage et de décodage très rapides.

Nous présenterons donc tout d'abord les fonctions de bases utilisées pour la décomposition étendu ici pour des dimensions quelconques d'images (sous-section 8.2.1), puis la création d'une pyramide gaussienne (sous-section 8.2.2) permettant enfin d'en arriver à la décomposition multi-fréquences en elle-même avec la création d'une pyramide laplacienne (sous-section 8.2.3).

Ce n'est qu'une fois ces étapes clairement définies que nous pourrons aborder la mélange d'images à proprement parler en exposant le cas présenté dans [42] (sous-section 8.2.4) avant de voir sa généralisation au cas d'un nombre quelconque d'images (sous-section 8.2.5) et, finalement, son application dans le cas de la création d'une texture pour un modèle 3D reconstruit à partir de vues multiples (sous-section 8.2.6).

8.2.1 Fonctions de réduction et d'expansion gaussiennes

Le calcul des pyramides gaussiennes et laplaciennes repose sur deux fonctions d'images de dimension n, la réduction et l'expansion gaussiennes. La réduction gaussienne donne en sortie une image dont toutes les dimensions sont (approximativement) la moitié de celles de l'image d'entrée, alors que pour l'expansion, les dimensions de l'image de sortie sont (approximativement) le double de celles d'entrée. Ces deux opérations utilisent un même noyau $\mathcal{N}$ de dimension n également et de taille 5^n (5 en 1D, 5×5 en 2D, $5 \times 5 \times 5$ en 3D,...) et qui est séparable par un produit de n noyaux 1D (par exemple, en 2D, $\mathcal{N}(i,j) = \mathcal{N}(i)\mathcal{N}(j)$). Nous ne nous attarderons donc à décrire les propriétés requises par le noyau 1D uniquement, les autres en découlant.

Le noyau $\mathcal{N}$ de taille 5 utilisé en 1D doit être :

- normalisé :
$$\sum_{i=-2}^{2} \mathcal{N}(i) = 1; \tag{8.6}$$
- symétrique :
$$\forall i \in [0;2], \mathcal{N}(i) = \mathcal{N}(-i), \tag{8.7}$$
on peut alors noter :
$$\begin{aligned} \mathcal{N}(0) &= a, \\ \mathcal{N}(1) = \mathcal{N}(-1) &= b, \\ \mathcal{N}(2) = \mathcal{N}(-2) &= c; \end{aligned} \tag{8.8}$$
- tel que la valeur de chaque élément d'entrée doit contribuer à part égale au calcul des valeurs des éléments de sortie, or chaque élément voit sa valeur pondérée (voir figure 8.7) :
  - soit une fois par le coefficient a et deux fois par le coefficient c,
  - soit deux fois par le coefficient b,

  nous avons donc :
$$a + 2c = 2b. \tag{8.9}$$

Ces contraintes sont satisfaites si l'on a :

$$\begin{aligned} \mathcal{N}(0) &= a, \\ \mathcal{N}(1) = \mathcal{N}(-1) &= \frac{1}{4}, \\ \mathcal{N}(2) = \mathcal{N}(-2) &= \frac{1}{4} - \frac{a}{2}. \end{aligned} \tag{8.10}$$

On remarque que la valeur de chaque pixel de l'image d'entrée participe pour moitié au calcul des valeurs des éléments de sortie dans le cas 1D. Dans la généralisation à une image de dimension n, ce rapport de participation devient $\frac{1}{2^n}$. Ceci est à mettre en relation avec le rapport entre la taille de l'image réduite et celle d'entrée qui est également de $\frac{1}{2^n}$.

En dimension 2, on obtient le noyau suivant :

$$\mathcal{N} = \begin{pmatrix} c^2 & bc & ac & bc & c^2 \\ bc & b^2 & ab & b^2 & bc \\ ac & ab & a^2 & ab & ac \\ bc & b^2 & ab & b^2 & bc \\ c^2 & bc & ac & bc & c^2 \end{pmatrix} = \begin{pmatrix} \frac{2a^2-2a+1}{16} & \frac{-2a+1}{16} & \frac{-2a^2+a}{4} & \frac{-2a+1}{16} & \frac{2a^2-2a+1}{16} \\ \frac{-2a+1}{16} & \frac{1}{16} & \frac{a}{4} & \frac{1}{16} & \frac{-2a+1}{16} \\ \frac{-2a^2+a}{4} & \frac{a}{4} & a^2 & \frac{a}{4} & \frac{-2a^2+a}{4} \\ \frac{-2a+1}{16} & \frac{1}{16} & \frac{a}{4} & \frac{1}{16} & \frac{-2a+1}{16} \\ \frac{2a^2-2a+1}{16} & \frac{-2a+1}{16} & \frac{-2a^2+a}{4} & \frac{-2a+1}{16} & \frac{2a^2-2a+1}{16} \end{pmatrix} \tag{8.11}$$

Il fut mis en évidence dans [41] que prendre $a = 0,4$ permettait d'obtenir la meilleure approximation d'une gaussienne.

Dans la suite, nous considérerons une image $\mathcal{I}$ de dimension n et de taille $T = (t_1, \ldots, t_n)$.

Remarque 8.1.
Il est à noter que, au bord de l'image, nous utiliserons les conditions aux limites de Neumann : la valeur retournée pour un élément en dehors des limites de l'image sera celle du pixel le plus proche dans l'image.

8.2.1.1 Fonction de réduction gaussienne

La réduction gaussienne de l'image $\mathcal{I}$, notée $RED(\mathcal{I})$, est une image $\mathcal{I}'$ de dimension n et de taille $T' = (t'_1, \ldots, t'_n)$ telle que :

$$\forall i \in [1; n], t'_i = \frac{t_i}{2} + 1. \tag{8.12}$$

Soient $\mathcal{N}$ le noyau gaussien de dimension n et $K = (k_1, \ldots, k_n)$ une coordonnée dans $\mathcal{N}$.

La valeur de tout élément de $\mathcal{I}' = RED(\mathcal{I})$ de coordonnées $X = (x_1, \ldots, x_n)$ est déterminée comme suit :

$$RED(\mathcal{I})(X) = \sum_{k_1=-2}^{2} \ldots \sum_{k_n=-2}^{2} \mathcal{N}(K) \times \mathcal{I}(X + K). \tag{8.13}$$

Par exemple, en dimension 2, on obtient :

$$RED(\mathcal{I})(X) = \sum_{k_1=-2}^{2} \sum_{k_2=-2}^{2} \mathcal{N}(K) \times \mathcal{I}(X + K). \tag{8.14}$$

On remarque que pour calculer les valeurs des éléments au bord de l'image réduite, il est nécessaire de connaître les valeurs d'éléments "en dehors" de l'image, d'où la nécessité des conditions au bord de Neumann (voir remarque 8.1) pour les interpoler.

On note $RED^k(\mathcal{I})$ le résultat de k réductions successives de l'image $\mathcal{I}$.

Une illustration de la réduction dans le cas 1D se trouve dans la figure 8.7.

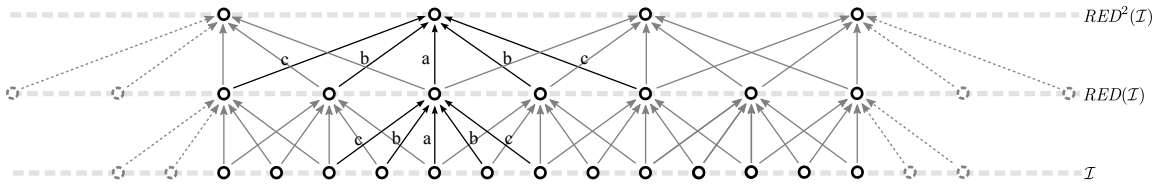


FIGURE 8.7 – Réductions 1D successives : les éléments représentés en pointillés gris sont les éléments interpolés (voir remarque 8.1) ; les flèches représentent les éléments d'un niveau pondérés par le noyau $\mathcal{N}$ intervenant dans le calcul d'un élément de la réduction (les coefficients ne sont pas tous représentés afin d'alléger la figure).

8.2.1.2 Fonction d'expansion gaussienne

L'expansion gaussienne de l'image $\mathcal{I}$, notée $EXP(\mathcal{I})$, est une image $\mathcal{I}'$ de dimension n et de taille $T' = (t'_1, \ldots, t'_n)$ telle que :

$$\forall i \in [1;n], t'_i = 2t_i - 1. \tag{8.15}$$

Soient $\mathcal{N}$ le noyau gaussien de dimension n et $K = (k_1, \ldots, k_n)$ une coordonnée dans $\mathcal{N}$.

La valeur de tout élément de $\mathcal{I}' = EXP(\mathcal{I})$ de coordonnées $X = (x_1, \ldots, x_n)$ est déterminée comme suit :

$$EXP(\mathcal{I})(X) = 2n \sum_{k_1=-2}^{2} \ldots \sum_{k_n=-2}^{2} \begin{cases} \mathcal{N}(K) \times \mathcal{I}(\frac{X-K}{2}) & \text{si } \forall i \in [1;n], \frac{x_i-k_i}{2} \text{ est un entier,} \\ 0 & \text{sinon.} \end{cases} \tag{8.16}$$

Par exemple, en dimension 2, on obtient :

$$EXP(\mathcal{I})(X) = 4 \sum_{k_1=-2}^{2} \sum_{k_2=-2}^{2} \begin{cases} \mathcal{N}(K) \times \mathcal{I}(\frac{X-K}{2}) & \text{si } \frac{x_1-k_1}{2} \text{ et } \frac{x_2-k_2}{2} \text{ sont des entiers,} \\ 0 & \text{sinon.} \end{cases} \tag{8.17}$$

Comme pour la réduction, on remarque que pour calculer les valeurs des éléments au bord de l'image étendue, il est nécessaire de connaître les valeurs d'éléments "en dehors" de l'image, d'où la nécessité des conditions au bord de Neumann (voir remarque 8.1) pour les interpoler.

On note $EXP^k(\mathcal{I})$ le résultat de k expansions successives de l'image $\mathcal{I}$.

Une illustration de l'expansion dans le cas 1D se trouve dans la figure 8.8.

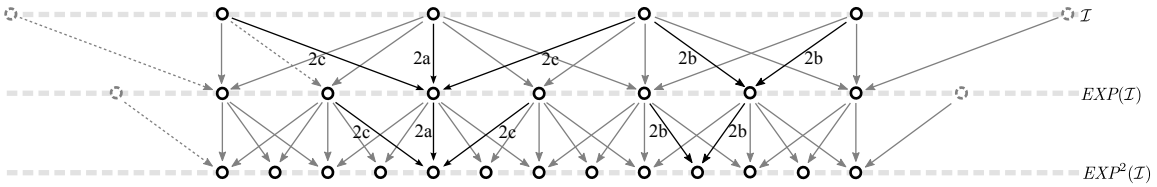


FIGURE 8.8 – Expansions 1D successives : les éléments représentés en pointillés gris sont les éléments interpolés (voir remarque 8.1) ; les flèches représentent les éléments d'un niveau coefficientés par le noyau $\mathcal{N}$ intervenant dans le calcul d'un élément de la réduction (les coefficients ne sont pas tous représentés afin d'alléger la figure).

8.2.1.3 Conditions sur la taille des images

Il est à noter que, dans [41], la taille d'une image à réduire ℓ fois doit être telle que :

$$\forall i \in [1;n], t_i = 2^\ell c_i + 1 \tag{8.18}$$

où, $\forall i \in [1;n]$, c_i est un entier naturel non nul.

Lorsque l'on procède à ℓ réductions successives d'une image puis à ℓ expansions successives, cette condition permet d'avoir pour résultat final une image ayant la même taille que l'image initiale.

Il est cependant possible de s'affranchir de cette contrainte en ne calculant, lors de l'expansion d'une image préalablement réduite, que les valeurs des éléments compris dans l'image d'origine. Ceci a pour conséquence de pouvoir nécessiter davantage d'éléments à

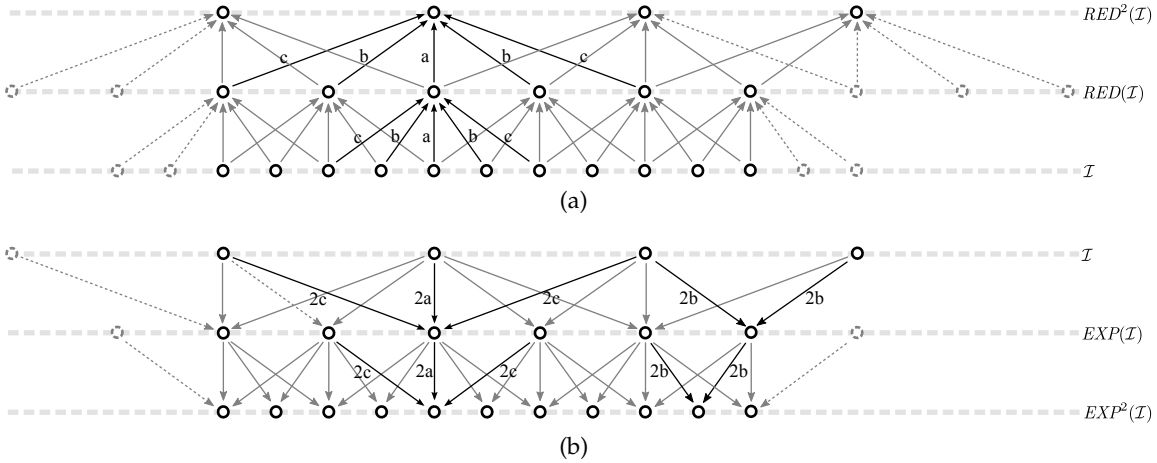


FIGURE 8.9 – Réductions (a) et expansions (b) successives en 1D pour des tailles d'images quelconques : les éléments représentés en pointillés gris sont les éléments interpolés (voir remarque 8.1) ; les flèches représentent les éléments d'un niveau coefficientés par le noyau $\mathcal{N}$ intervenant dans le calcul d'un élément de la réduction (les coefficients ne sont pas tous représentés afin d'alléger la figure).

interpoler hors de l'image (voir remarque 8.1) lors des réductions (voir figure 8.9a) mais moins lors des expansions (voir figure 8.9b).

Une autre façon de procéder serait d'agrandir l'image d'entrée en rajoutant le nombre minimum d'éléments permettant d'avoir une taille répondant à la condition donnée ci-avant. Les valeurs des éléments ajoutés seraient alors calculés selon les conditions aux bords de Neumann (voir remarque 8.1). Cependant, de cette manière, le nombre de calculs à faire pour obtenir les pyramides gaussienne et laplacienne serait accru.

8.2.2 Pyramide gaussienne

Soit ℓ le nombre de niveaux de la pyramide. Soit une image d'entrée $\mathcal{I}$.

On note $\mathcal{G}_i(\mathcal{I})$, avec $i \in [0; \ell - 1]$, l'image du niveau i de la pyramide gaussienne de l'image $\mathcal{I}$.

L'image d'entrée est le niveau 0 de la pyramide gaussienne :

$$\mathcal{G}_0(\mathcal{I}) = \mathcal{I} \tag{8.19}$$

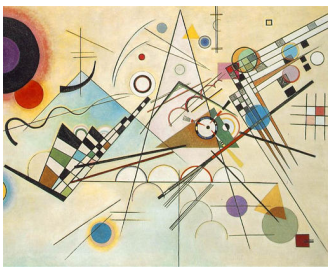
Les niveaux supérieurs se calculent comme suit :

$$\begin{aligned} \forall k \in [1; \ell - 1], \mathcal{G}_k(\mathcal{I}) &= RED^k(\mathcal{I}) \\ &= RED(\mathcal{G}_{k-1}(\mathcal{I})) \end{aligned} \tag{8.20}$$

Autrement dit, l'image d'un niveau de la pyramide gaussienne est obtenu par une réduction de son niveau inférieur.

Un exemple de pyramide gaussienne (accompagnée des expansions de ses différents niveaux pour une meilleure visualisation) se trouve dans la figure 8.10.

On remarque que les niveaux de la pyramide gaussienne constituent un filtre de fréquences passe-bas : plus on monte dans les niveaux de la pyramide, plus le seuil de fréquence, au-dessous duquel les fréquences de l'image sont conservées, diminue.



(a) $\mathcal{I} = \mathcal{G}_0(\mathcal{I})$

(b) $\mathcal{G}_1(\mathcal{I})$

(c) $EXP(\mathcal{G}_1(\mathcal{I}))$

(d) $\mathcal{G}_2(\mathcal{I})$

(e) $EXP^2(\mathcal{G}_2(\mathcal{I}))$

(f) $\mathcal{G}_3(\mathcal{I})$

(g) $EXP^3(\mathcal{G}_3(\mathcal{I}))$

(h) $\mathcal{G}_4(\mathcal{I})$

(i) $EXP^4(\mathcal{G}_4(\mathcal{I}))$

(j) $\mathcal{G}_5(\mathcal{I})$

(k) $EXP^5(\mathcal{G}_5(\mathcal{I}))$

(l) $\mathcal{G}_6(\mathcal{I})$

(m) $EXP^6(\mathcal{G}_6(\mathcal{I}))$

FIGURE 8.10 – Pyramide gaussienne de 7 niveaux d'une image 2D.

8.2.3 Pyramide laplacienne

De manière similaire au cas de la pyramide gaussienne, on note $\mathcal{L}_i(\mathcal{I})$, avec $i \in [0; \ell - 1]$, l'image du niveau i de la pyramide laplacienne de l'image $\mathcal{I}$.

L'image du niveau le plus élevé est l'image de niveau équivalent dans la pyramide gaussienne de l'image :

$$\begin{aligned} \mathcal{L}_{\ell-1}(\mathcal{I}) &= \mathcal{G}_{\ell-1}(\mathcal{I}) \\ &= RED^k(\mathcal{I}) \end{aligned} \tag{8.21}$$

Les niveaux inférieurs se calculent comme suit :

$$\forall k \in [0; \ell - 2], \mathcal{L}_k(\mathcal{I}) = \mathcal{G}_k(\mathcal{I}) - EXP(\mathcal{G}_{k+1}(\mathcal{I})) \tag{8.22}$$

Autrement dit, l'image d'un niveau de la pyramide laplacienne est obtenue à partir du niveau équivalent de la pyramide gaussienne auquel on soustrait une expansion de son niveau supérieur.

Une illustration de cette construction se trouve dans la figure 8.11

Un exemple de pyramide laplacienne (accompagnée des expansions de ses différents niveaux pour une meilleure visualisation) se trouve dans la figure 8.12.

On remarque que, pour chaque niveau de la pyramide laplacienne, seule une bande de fréquences de l'image d'origine est conservée. Plus on avance dans les niveaux de la pyramide, plus les fréquences conservées diminuent. Celles-ci correspondent aux fréquences comprises entre les seuils des deux niveaux la pyramide gaussienne qui ont servi au calcul du niveau de la pyramide laplacienne. Le dernier niveau, quant à lui, contient toutes les fréquences restantes, qui sont donc les plus basses de la pyramide laplacienne.

Il est à noter que, par construction d'une pyramide laplacienne, on peut retrouver l'image d'origine à partir de celle-ci. Il suffit de sommer les différents niveaux de la pyramide après avoir procédé à leur expansion de sorte à avoir une taille identique à celle du niveau le plus bas :

$$\mathcal{I} = \mathcal{L}_0(\mathcal{I}) + \sum_{k=1}^{\ell-1} EXP^k \mathcal{L}_k(\mathcal{I}) \tag{8.23}$$

Cette opération somme ainsi les différentes bandes de fréquences de l'image initiale qui avaient été séparées dans les différents niveaux de la pyramide laplacienne.

Afin de diminuer le nombre d'expansions à exécuter, cette reconstruction peut être reformulée comme suit :

$$\mathcal{I} = \mathcal{L}_0(\mathcal{I}) + EXP(\mathcal{L}_1(\mathcal{I}) + EXP(\mathcal{L}_2(\mathcal{I}) + EXP(\ldots + EXP(\mathcal{L}_{\ell-1}(\mathcal{I}))\ldots))) \tag{8.24}$$

Cela revient à faire l'opération inverse à la construction de la pyramide laplacienne en reconstruisant la pyramide gaussienne. Pour cela, on somme, pour chaque niveau, le niveau correspondant de la pyramide laplacienne et le niveau inférieur de la pyramide gaussienne.

Une illustration de cette méthode se trouve dans la figure 8.13.

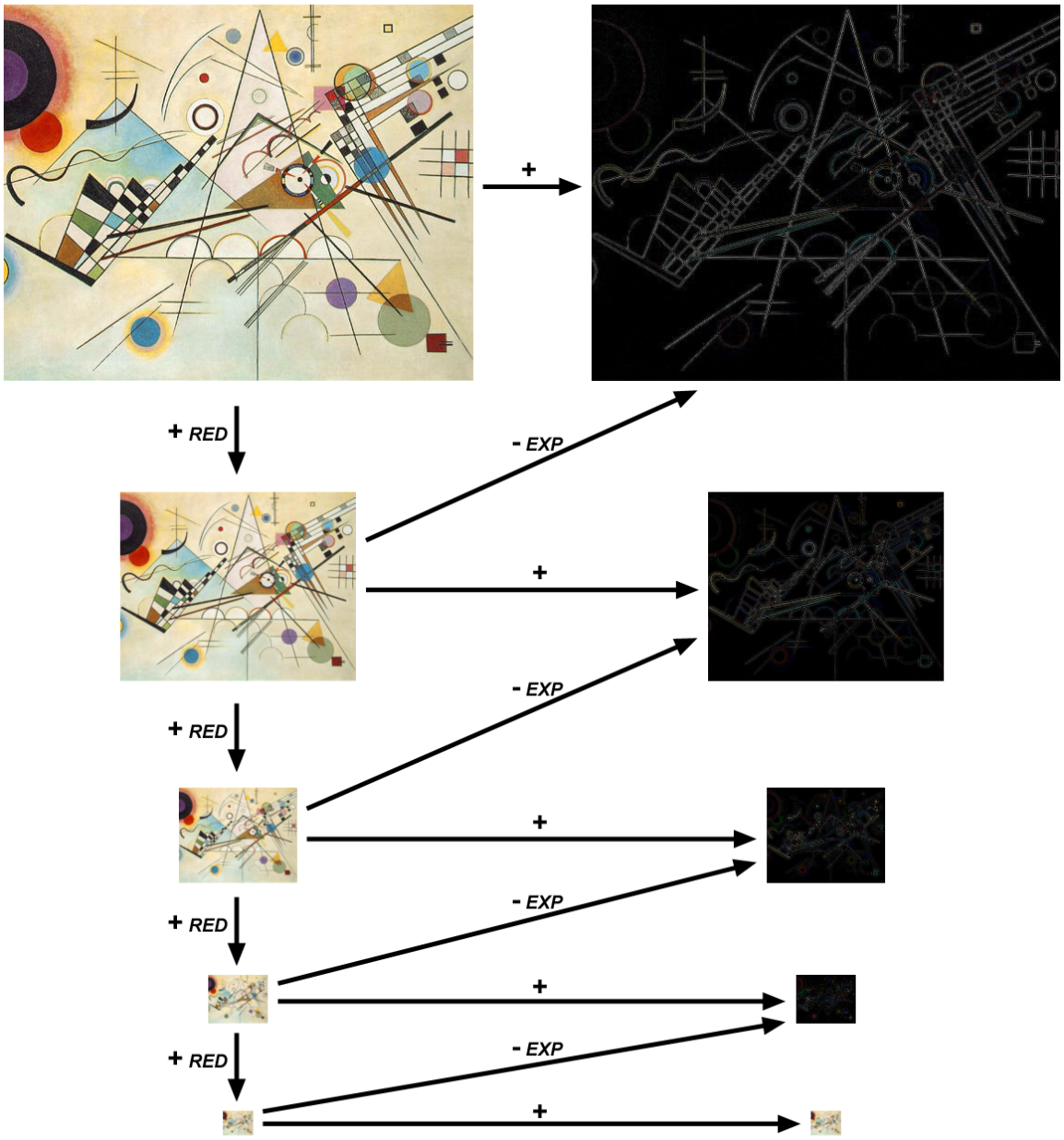
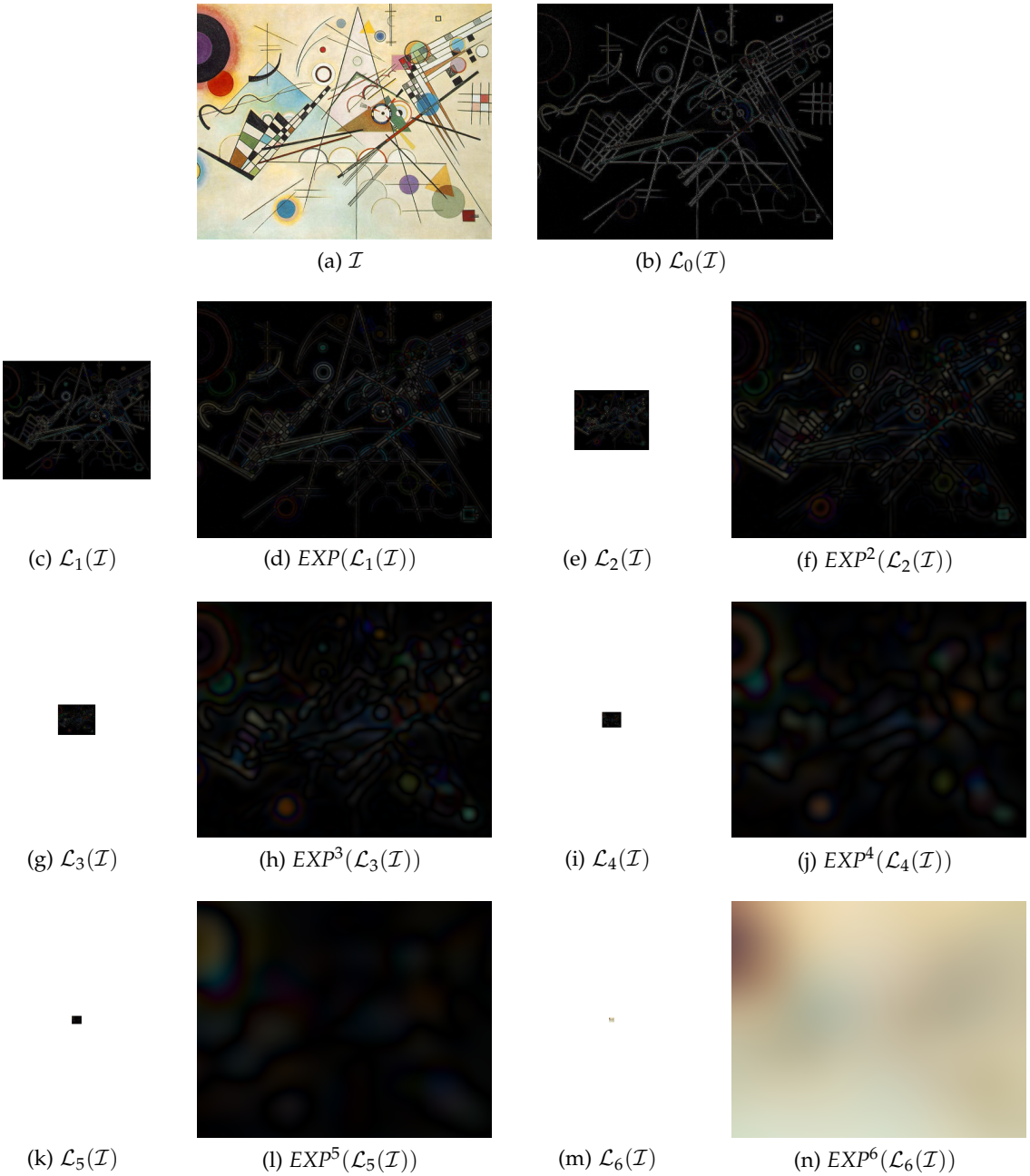


FIGURE 8.11 – Construction de pyramides gaussienne et laplacienne de cinq niveaux.



(a) $\mathcal{I}$ (b) $\mathcal{L}_0(\mathcal{I})$

(c) $\mathcal{L}_1(\mathcal{I})$ (d) $EXP(\mathcal{L}_1(\mathcal{I}))$ (e) $\mathcal{L}_2(\mathcal{I})$ (f) $EXP^2(\mathcal{L}_2(\mathcal{I}))$

(g) $\mathcal{L}_3(\mathcal{I})$ (h) $EXP^3(\mathcal{L}_3(\mathcal{I}))$ (i) $\mathcal{L}_4(\mathcal{I})$ (j) $EXP^4(\mathcal{L}_4(\mathcal{I}))$

(k) $\mathcal{L}_5(\mathcal{I})$ (l) $EXP^5(\mathcal{L}_5(\mathcal{I}))$ (m) $\mathcal{L}_6(\mathcal{I})$ (n) $EXP^6(\mathcal{L}_6(\mathcal{I}))$

FIGURE 8.12 – Pyramide laplacienne de 7 niveaux d'une image 2D.

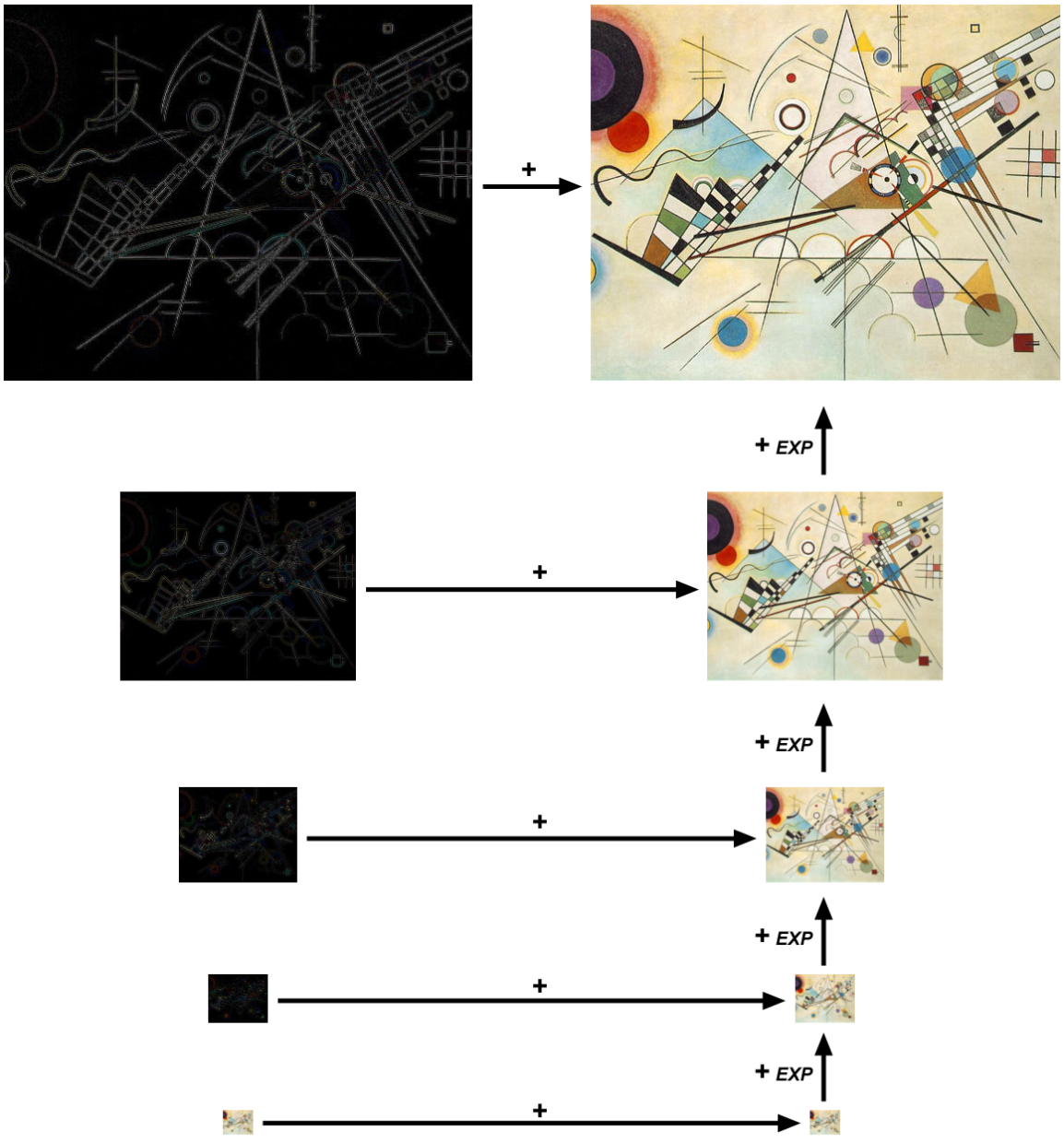


FIGURE 8.13 – Reconstruction de l'image originale et de sa pyramide gaussienne à partir de sa pyramide laplacienne de cinq niveaux.

8.2.4 Mélange de deux images

Le type de mélange décrit ci-dessous est celui effectué à l'aide d'un masque binaire dans [42].

Soient $\mathcal{I}_A$ et $\mathcal{I}_B$ les deux images de même taille à mélanger. Soit M le masque de mélange entre les images $\mathcal{I}_A$ et $\mathcal{I}_B$ (de même taille que celles-ci) tel que M vaut 1 là où on souhaite avoir l'image $\mathcal{I}_A$ et 0 là où on souhaite avoir l'image $\mathcal{I}_B$. Soit ℓ le nombre de niveaux des pyramides à construire.

Le mélange se fait comme suit :

1. Construire les pyramides laplaciennes de ℓ niveaux $\mathcal{L}^A$ et $\mathcal{L}^B$ pour les images $\mathcal{I}_A$ et $\mathcal{I}_B$.
2. Construire la pyramide gaussienne de ℓ niveaux $\mathcal{G}^M$ pour le masque M.
3. Construire une pyramide de ℓ niveaux $\mathcal{L}^R$ (avec des images de taille identique à celles de $\mathcal{L}^A$, $\mathcal{L}^B$ et $\mathcal{G}^M$ pour chaque niveau) combinée entre $\mathcal{L}^A$ et $\mathcal{L}^B$ coefficientées par $\mathcal{G}^M$. Ainsi, pour tout niveau $k \in [0; \ell - 1]$ et toutes coordonnées X comprises dans l'image $\mathcal{L}^R_k$:

$$\mathcal{L}^R_k(X) = \mathcal{G}^M_k(X)\mathcal{L}^A_k(X) + (1 - \mathcal{G}^M_k(X))\mathcal{L}^B_k(X). \tag{8.25}$$

4. Construire l'image résultat en sommant les niveaux de $\mathcal{L}^R$ après expansion comme pour une pyramide laplacienne classique (voir sous-section 8.2.3 et figure 8.13).

Par cette méthode, les plus hautes fréquences ne sont ainsi mélangées que sur d'étroites bandes autour de la frontière entre les régions délimitées par le masque, alors que les plus basses le sont sur des bandes plus larges, mesurant $2^{\ell-1}$ pixels pour les dernières.

Deux exemples de mélanges d'images 2D suivant cette méthode se trouvent dans les figures 8.14 et 8.16. De plus, une illustration de son déroulement pour le premier exemple se trouve dans la figure 8.15.

On remarque que le mélange avec dégradé par canal alpha (figures 8.14e et 8.16e) produit des artefacts de type fantôme. Dans la figure 8.16e, certains copeaux de chocolat proches de la frontière entre les deux images d'entrée apparaissent comme "transparents". Ceci est dû au fait que la transition est faite sans tenir compte des fréquences. En effet, dans le mélange d'images multi-fréquences (figure 8.16f), même si les couleurs de certains copeaux est altérée leurs contours restent nets puisque celui-ci est contenu dans les hautes fréquences de l'image qui ne sont donc mélangées que sur une étroite bande de part et d'autre de la frontière.

8.2.5 Mélange de plus de deux images

Devant la limitation au mélange de seulement deux images qu'offre la méthode décrite dans la section 8.2.4, nous l'avons modifié afin de l'étendre à un nombre quelconque d'images.

Soient n images $\mathcal{I}_0$ à $\mathcal{I}_{n-1}$ de même taille à mélanger. Soit M le masque de mélange entre les images $\mathcal{I}_1$ à $\mathcal{I}_{n-1}$ (de même taille que celles-ci) tel que M vaut i avec $i \in [0; n-1]$ là où on souhaite avoir l'image $\mathcal{I}_i$. Soit ℓ le nombre de niveaux des pyramides à construire.

Le mélange se fait comme suit :

1. Pour chaque image $\mathcal{I}_i$, construire les pyramides laplaciennes de ℓ niveaux $\mathcal{L}^i$.

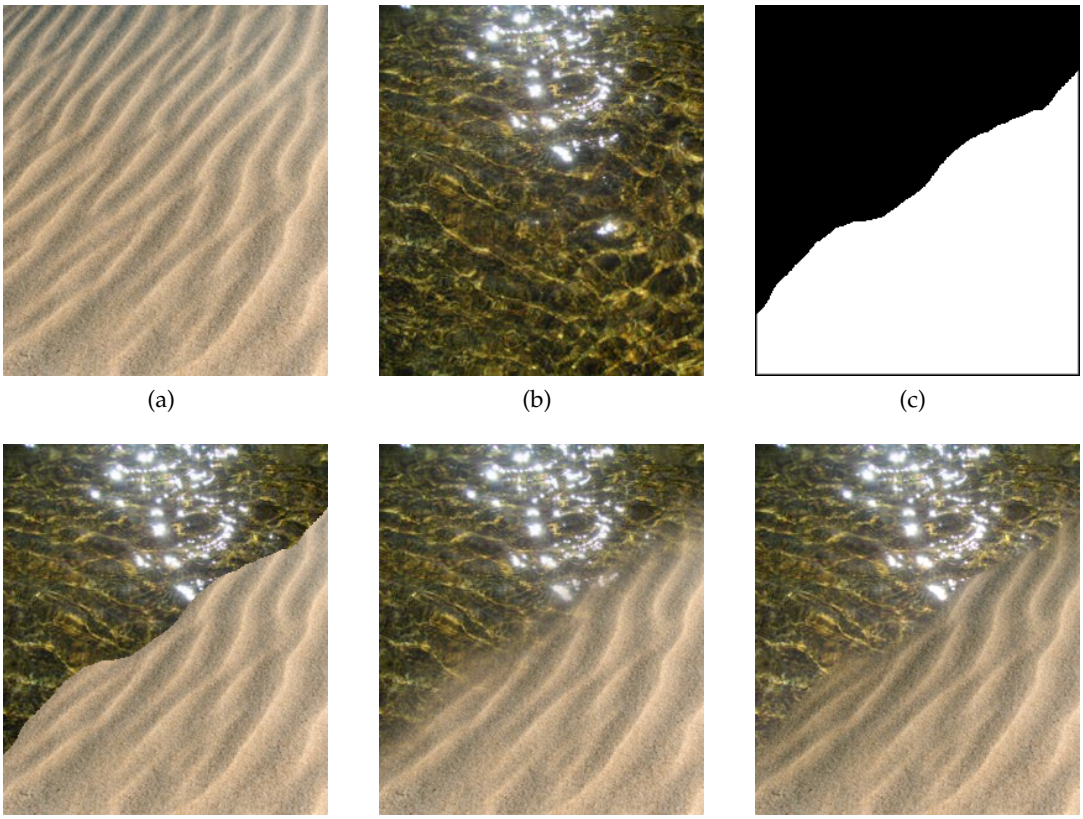


FIGURE 8.14 – Mélange d'images à 7 niveaux avec : (a,b) deux images d'entrée ; (c) masque de mélange des images ; (d) mélange "direct" ; (e) mélange avec dégradé par canal alpha ; (f) mélange multi-fréquences.

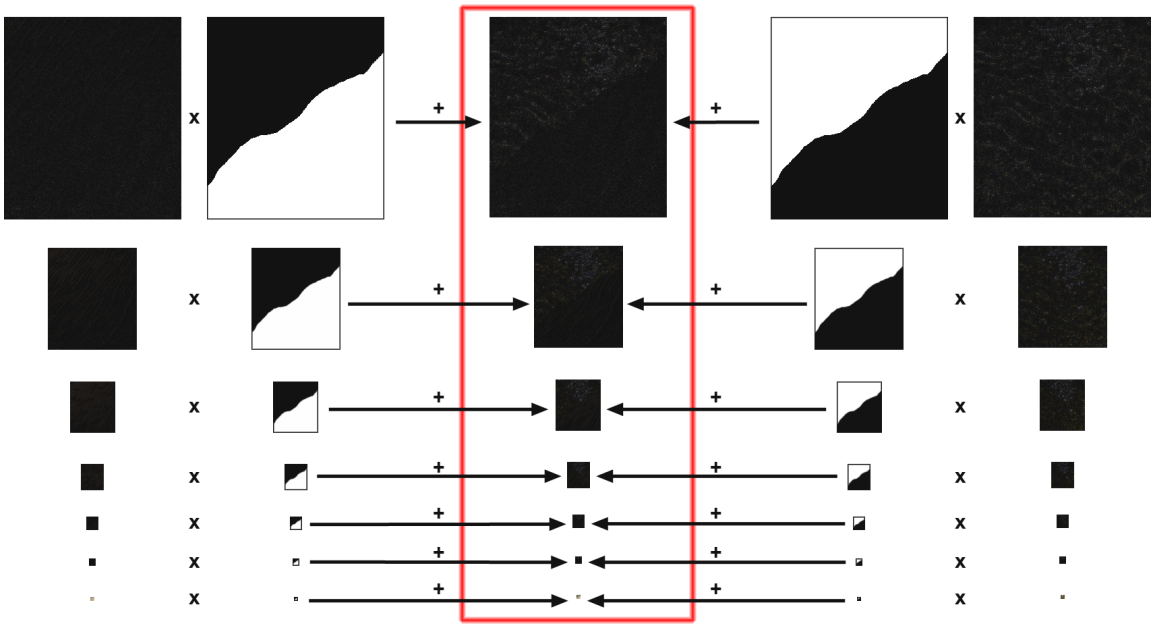
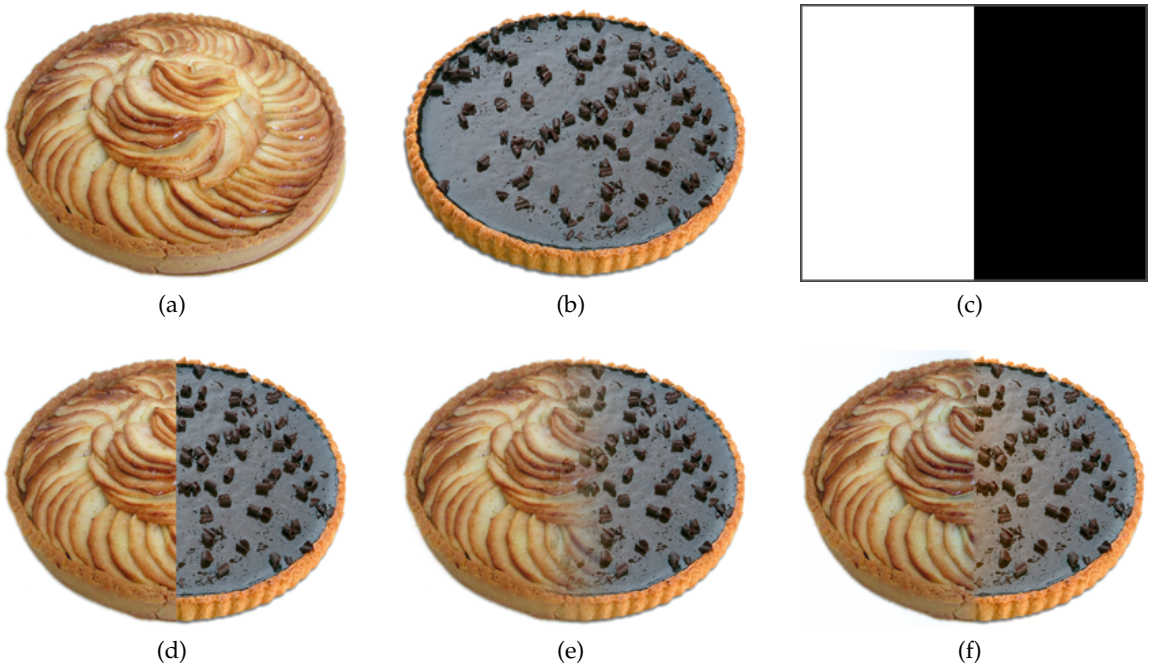


FIGURE 8.15 – Déroulement du mélange des images de la figure 8.14 avec, pour chaque colonne, de gauche à droite, les 7 niveaux de : la pyramide laplacienne de l'image 8.14a, la pyramide gaussienne du masque 8.14c, la pyramide laplacienne constituant le mélange des images 8.14a et 8.14b, la pyramide gaussienne du masque 8.14c inversé et, enfin, la pyramide laplacienne de l'image 8.14b.



(a) (b) (c)

(d) (e) (f)

FIGURE 8.16 – Mélange d'images à 7 niveaux avec : (a,b) deux images d'entrée ; (c) masque de mélange des images ; (d) mélange "direct" ; (e) mélange avec dégradé par canal alpha ; (f) mélange multi-fréquences.

2. Pour chaque image $\mathcal{I}_i$, créer le masque binaire de mélange M_i lui correspondant à partir de M tel que M_i vaut 1 là où M vaut i et 0 ailleurs.
3. Pour chaque masque M_i, construire les pyramides gaussiennes de ℓ niveaux $\mathcal{G}^i$.
4. Construire une pyramide de ℓ niveaux $\mathcal{L}^R$ (avec des images de taille identique à celles des différentes pyramides laplaciennes $\mathcal{L}^i$ pour chaque niveau) combinée entre les différentes $\mathcal{L}^i$ coefficientées par $\mathcal{G}^i$. Ainsi, pour tout niveau $k \in [0; \ell - 1]$ et toutes coordonnées X comprises dans l'image $\mathcal{L}^R_k$:

$$\mathcal{L}^R_k(X) = \frac{\sum_{i=1}^{n} \mathcal{G}^i_k(X)\mathcal{L}^i_k(X)}{\sum_{i=1}^{n} \mathcal{G}^i_k(X)} \tag{8.26}$$

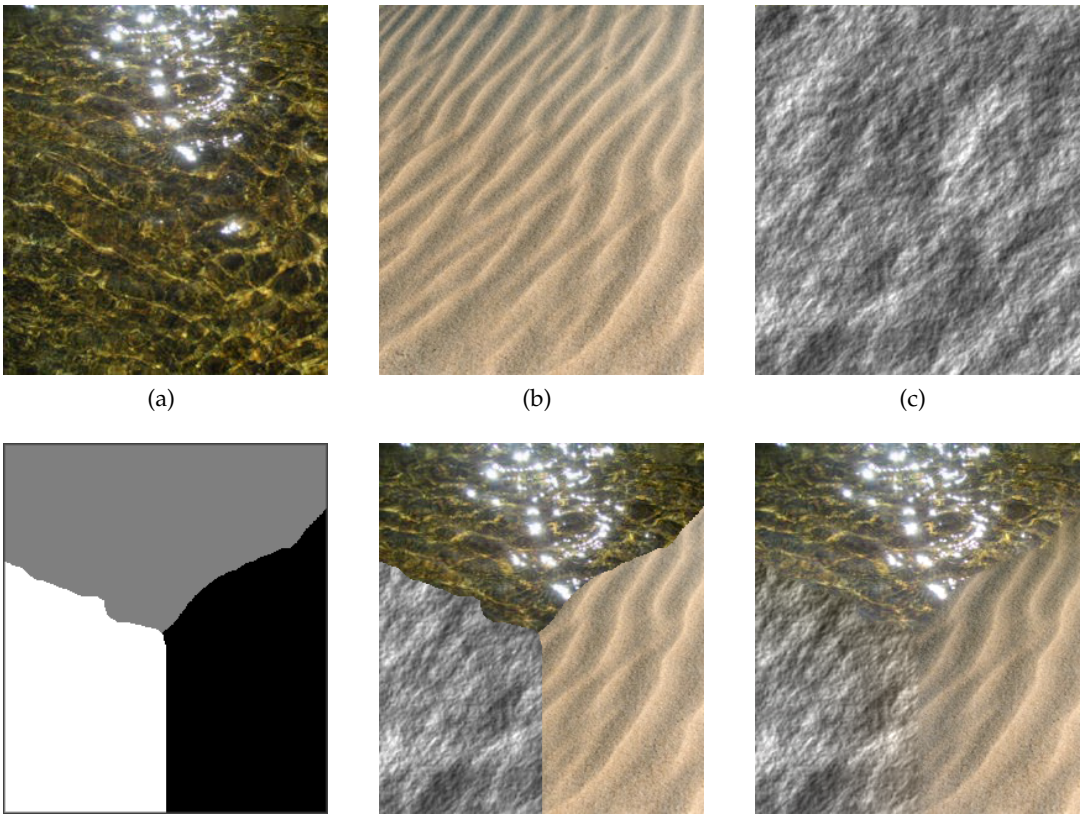
5. Construire l'image résultat en sommant les niveaux de $\mathcal{L}^R$ après expansion comme pour une pyramide laplacienne classique (voir sous-section 8.2.3 et figure 8.13).

Un exemple de mélange de trois images 2D suivant cette méthode se trouve dans la figure 8.17.

8.2.6 Mélange de textures sur modèle 3D reconstruit

Nous allons maintenant utiliser le mélange multi-fréquences pour un nombre quelconque d'images présenté dans la section précédente pour faire disparaître les jointures de la texture des modèles 3D.

Le problème ne concerne maintenant plus des images dont les coordonnées des pixels coïncident exactement, mais des projections d'images sur la surface du maillage. Il nous faut alors déterminer, pour chaque point X de la surface du maillage, la couleur $C(X)$.



(a) (b) (c)

(d) (e) (f)

FIGURE 8.17 – Mélange de plusieurs images avec : (a,b,c) trois images d'entrée ; (d) masque de mélange des images ; (e) mélange "directe" ; (f) mélange multi-fréquences.

Cette couleur est une combinaison des différentes bandes de fréquences des différentes images dans lesquelles X est visible.

Il faut tout d'abord construire les éléments suivants :

- pour toute image d'entrée $\mathcal{I}_i$, construire les pyramides laplaciennes de ℓ niveaux $\mathcal{L}^i$;
- pour toute image $\mathcal{I}_i$, créer le masque binaire M_i représentant les régions de l'image utilisées dans la projection de texture sur le maillage (i.e. un pixel est à 1 s'il est utilisé dans la texture du maillage, à 0 sinon) puis construire sa pyramide gaussienne de ℓ niveaux $\mathcal{G}^i$.

On définit une nouvelle fonction de projection Π_k^i de l'espace 3D dans le $k^{\text{ième}}$ niveau de la pyramide (gaussienne ou laplacienne) de l'image I_i.

Il suffit ensuite de procéder au même type de reconstruction que pour le cas 2D en prenant en compte les changements de coordonnées par les différentes fonction de projection. La couleur de la texture en chaque point de la surface du maillage se calcule alors comme suit :

$$C(X) = \sum_{k=0}^{\ell-1} \frac{\sum_{i=1}^{n} w_k^i(X)\mathcal{L}_k^i(\Pi_k^i(X))}{\sum_{i=1}^{n} w_k^i(X)} \tag{8.27}$$

où

$$w_k^i(X) = \begin{cases} \mathcal{G}_k^i(\Pi_k^i(X)) & \text{si } X \text{ est visible dans l'image } I_i \text{ (i.e. } M_i(\Pi^i(X)) = 1, \\ 0 & \text{sinon.} \end{cases} \tag{8.28}$$

D'un point de vue pratique, on discrétise C dans le domaine des images d'entrée en procédant au calcul des nouvelles valeurs de l'ensemble des pixels utilisés dans la texture (i.e. les pixels étiquetés comme tels dans chaque masque M_i).

Le résultat du mélange d'image est stocké dans une nouvelle série d'images $I'_1, \ldots, I'_n$ tel que pour tout point X de la facette f_j, la couleur $C(X)$ est stockée dans l'image I'_{x_j} au pixel de coordonnées $\Pi_{x_j}(x)$. Il est à noter que, dans chaque image I'_i, seuls les pixels ayant des coordonnées dans la région valant 1 du masque correspondant M_i ont une valeur de sauvegardée. Le reste de l'image n'étant pas utilisée, la valeur des pixels n'a pas d'importance et peut donc être laissée par défaut à 0.

Cette dernière série d'images sera utilisée par la suite pour construire l'atlas de texture.

8.3 Atlas de texture

Une fois notre texture calculée, il s'avère intéressant de la sauvegarder de sorte à pouvoir la visualiser sans repasser par les phases de calcul précédentes.

Créer de nouvelles vues à partir de la texture obtenue est possible avec plusieurs passes de texturation projective comme dans [68]. Cependant, la création d'un atlas, qui consiste en une seule image contenant toutes les régions des vues à projeter sur le maillage s'avère plus intéressant. En effet, avoir une unique carte de texture permet d'avoir un rendu plus rapide ainsi que la sauvegarde en un format de fichier 3D standard.

Pour créer un tel atlas, nous appliquons tout d'abord une dilatation morphologique aux masques M_i avec un élément structurant carré de quelques pixels de côté. Cette bande de pixels autour des régions utilisées dans la texture du maillage est nécessaire pour la magnification automatique de la texture lors de la phase de rendu.

Nous décomposons ensuite chaque image en composantes connexes d'après leurs masques afin de ne conserver que les régions dilatées qui sont utilisées dans la texture du maillage. Nous regroupons ensuite ces fragments de textures dans une seule et même image de taille suffisante pour tout contenir, qui formera notre atlas, grâce à une stratégie de type *first-fit decreasing* : les fragments de textures sont pris par ordre décroissant de taille et positionnés à la première place disponible (i.e. qui ne recouvre aucun fragment déjà placé) dans le sens de balayage vidéo de l'atlas. Enfin, nous transférons les coordonnées de la texture des images d'entrées vers les différents fragments correspondants de notre atlas. Le résultat final de notre algorithme est ainsi compatible avec la majeure partie des programmes de visualisation d'objets 3D.

8.4 Resultats

Notre implémentation de génération d'atlas de texture sans jointure fait appel à la programmation sur GPU (unité de processeur graphique, *graphics processing unit* en anglais), et plus précisément à la carte de profondeurs, pour calculer la visibilité des facettes pour chacun des différents points de vue.

Nous utilisons également la librairie de minimisation par coupes minimales développée par Olga Veksler (http://www.csd.uwo.ca/~olga/code.html) et by Vladimir Kolmogorov (http://www.adastral.ucl.ac.uk/~vladkolm/software.html) pour l'optimisation de notre partition de la surface du maillage.

Les figures 8.18 et 8.19 présentent des exemples de masques des régions d'images utilisées dans la texture pour les différentes vues correspondantes après l'optimisation de la partition telle que décrite dans la section 8.1.

Les figures 8.20 et 8.21 présentent une vue du maillage 3D avec trois projections de textures différentes :

a) l'approche "naïve" consistant à projeter indépendamment sur chaque facette l'image qui lui offre la meilleure résolution de texture ;
b) l'approche par optimisation de partition, consistant uniquement en la partie de notre méthode décrite dans la section 8.1 (pas de correction de couleurs) ;
c) notre approche complète, qui ajoute le mélange de texture décrit dans la section 8.2 à l'approche par optimisation de partition ci-dessus.

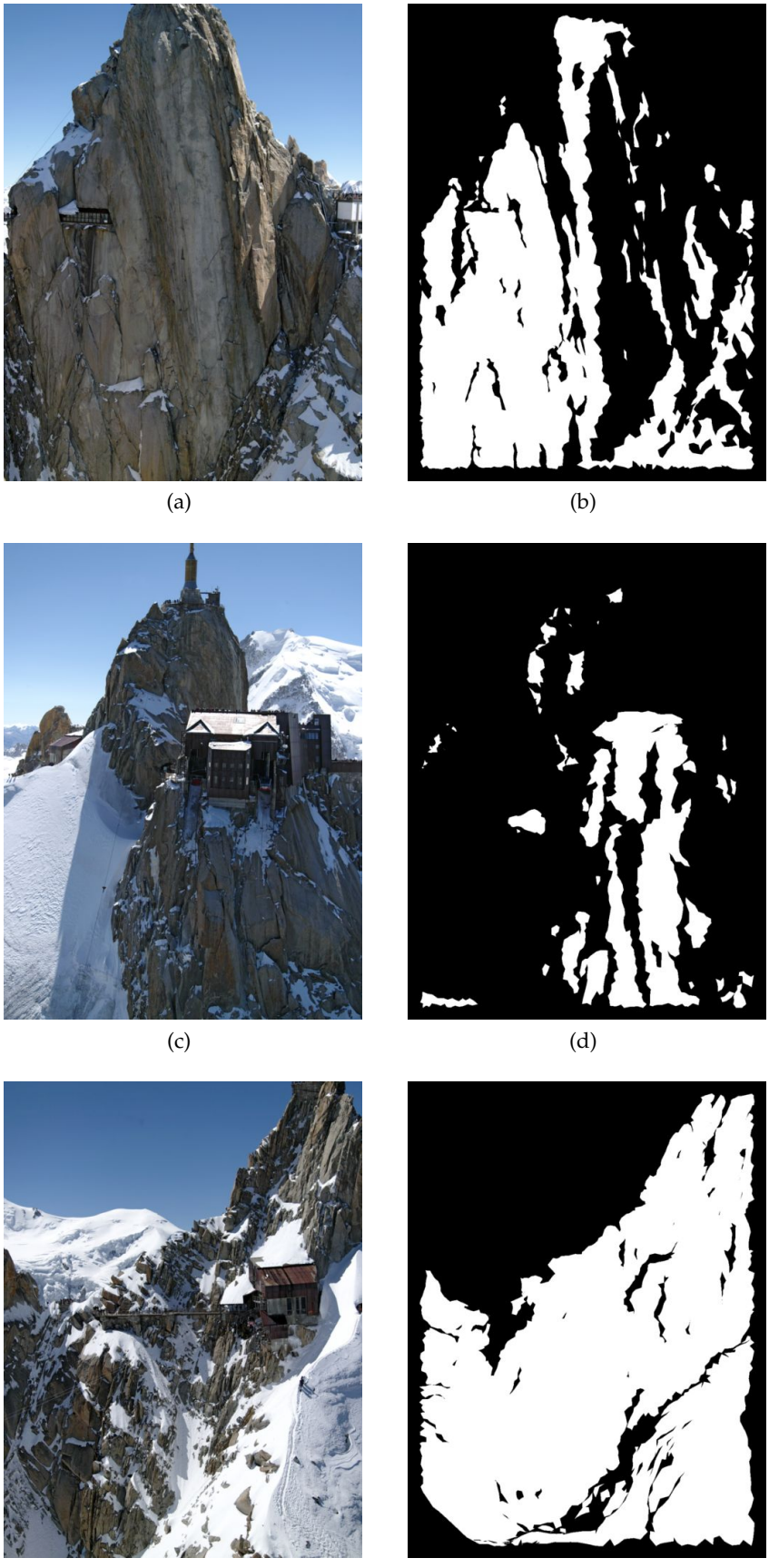
Les figures 8.22 et 8.23 présentent une vue du maillage 3D avec les projections des partitions de texture "naïves" et opitmisées.

Enfin, les atlas correspondants obtenus avec notre méthode sont présentés dans la figure 8.24.

On observe une amélioration du rendu de la texture entre chacune des étapes présentées dans les figures 8.20 et 8.21. Ceci s'observe notamment au niveau de la roche dans la figure 8.20 et au niveau du sol ou du mur du fond dans la figure 8.21.

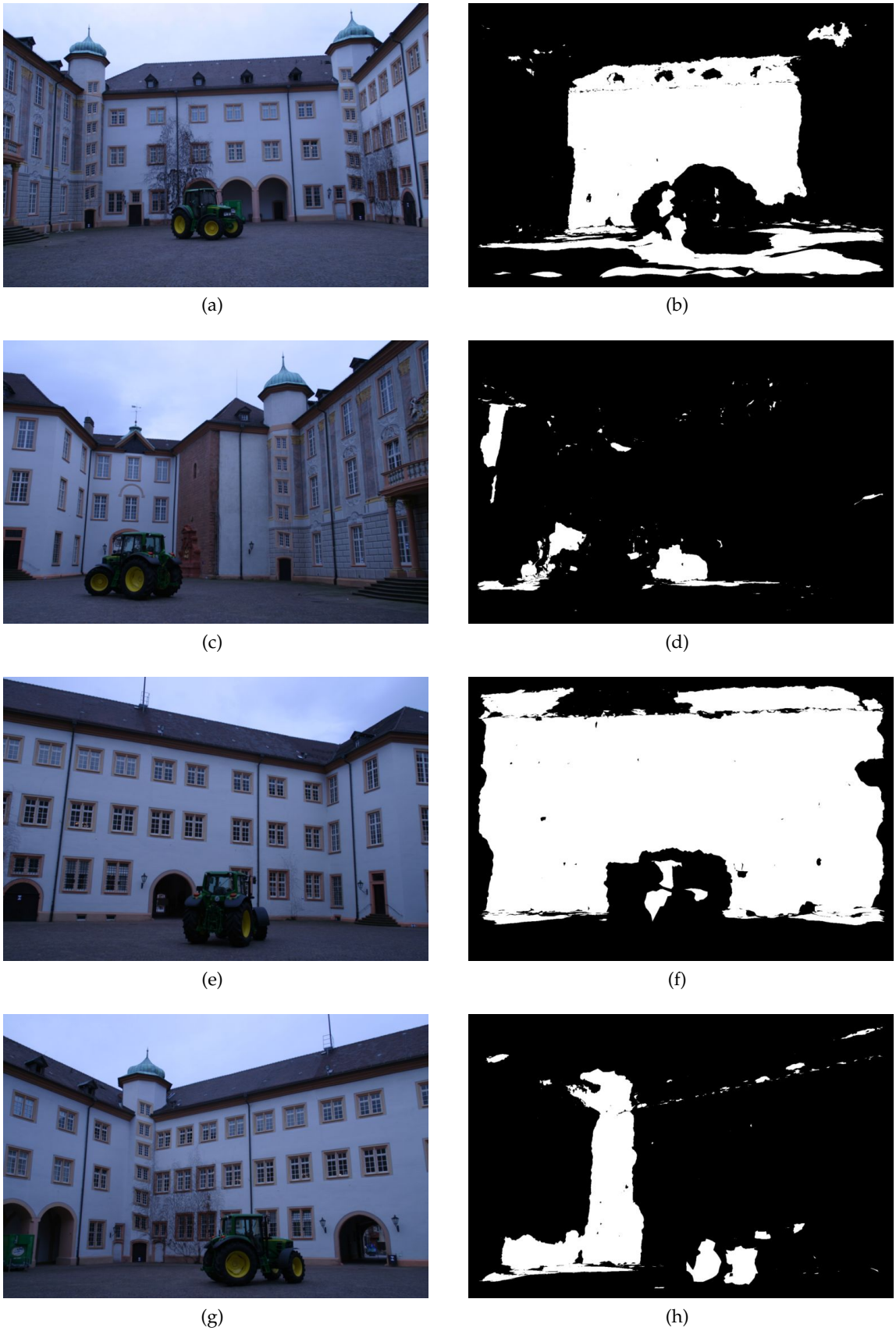
De nombreuses jointures sont visibles dans l'approche "naïve" (cas (a)). Leur nombre se voit grandement réduit en ajoutant l'optimisation de partition (cas (b)) mais il en demeure encore certaines de visibles dû aux trop grandes variations de luminosité et de prises de vue. Celles-ci disparaissent complètement avec notre méthode complète (cas (c)).

Des vidéos et des vues 3D interactives de nos résultats sont disponibles à cette adresse : http://certis.enpc.fr/~allene/research-3Dblending.html.



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

FIGURE 8.18 – (a,c,e) Photos de l'Aiguille du Midi. (b,d,f) Masques respectifs des régions utilisées dans la texture du maillage 3D.



(a) (b) (c) (d) (e) (f) (g) (h)

FIGURE 8.19 – (a,c,e) Photos du château d'Ettlingen. (b,d,f) Masques respectifs des régions utilisées dans la texture du maillage 3D.

(a)

(b)

(c)

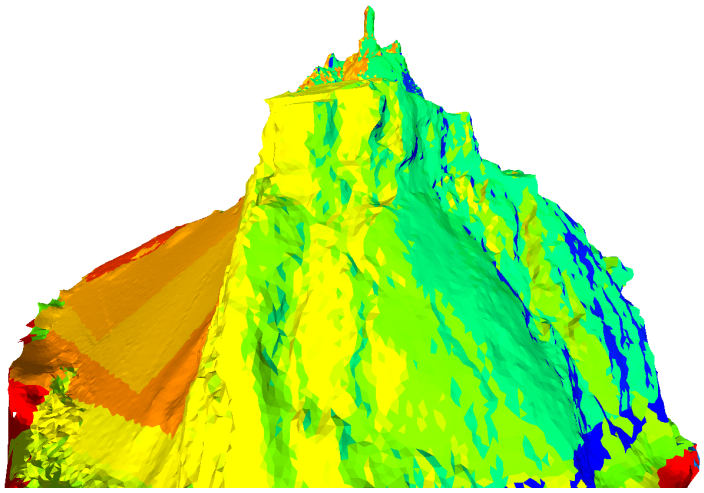
FIGURE 8.20 – Reconstruction 3D de l'Aiguille du Midi avec les textures générées par : (a) l'approche naïve ; (b) l'approche par optimisation de partition ; (c) notre méthode complète.

(a)

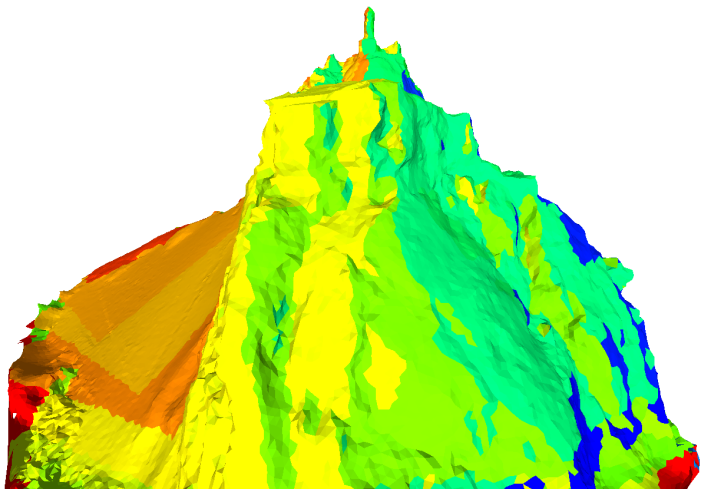
(b)

(c)

FIGURE 8.21 – Reconstruction 3D du château d'Ettlingen avec les textures générées par : (a) l'approche naïve ; (b) l'approche par optimisation de partition ; (c) notre méthode complète. Les "tâches" blanches sont en fait des "trous" dans le maillage liés à des imprécisions lors de sa construction à partir des différentes vues.

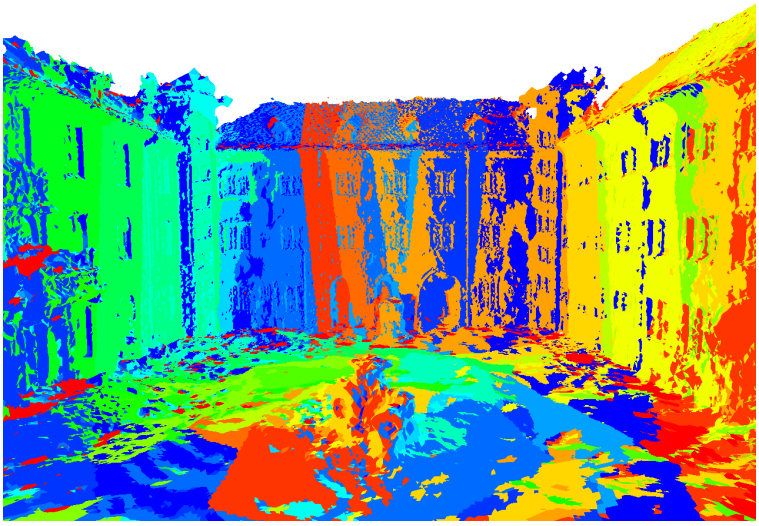


(a)

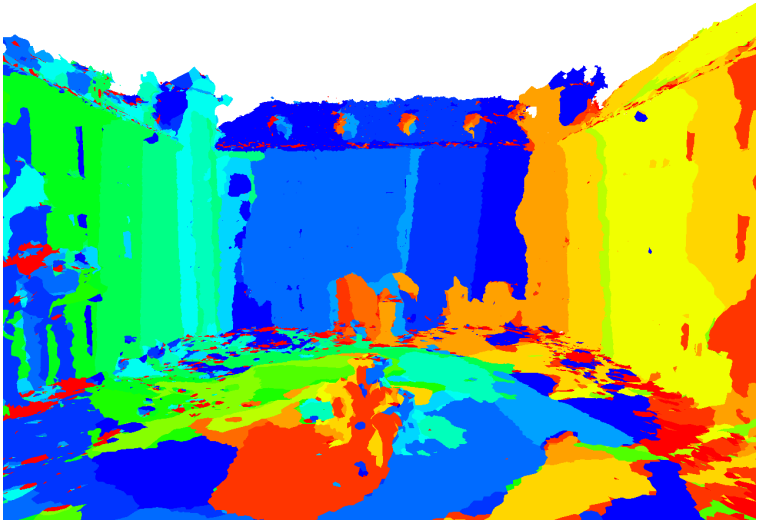


(b)

FIGURE 8.22 – Reconstruction 3D de l'Aiguille du Midi avec la partition des textures générées par : (a) l'approche naïve ; (b) l'approche par optimisation de partition.

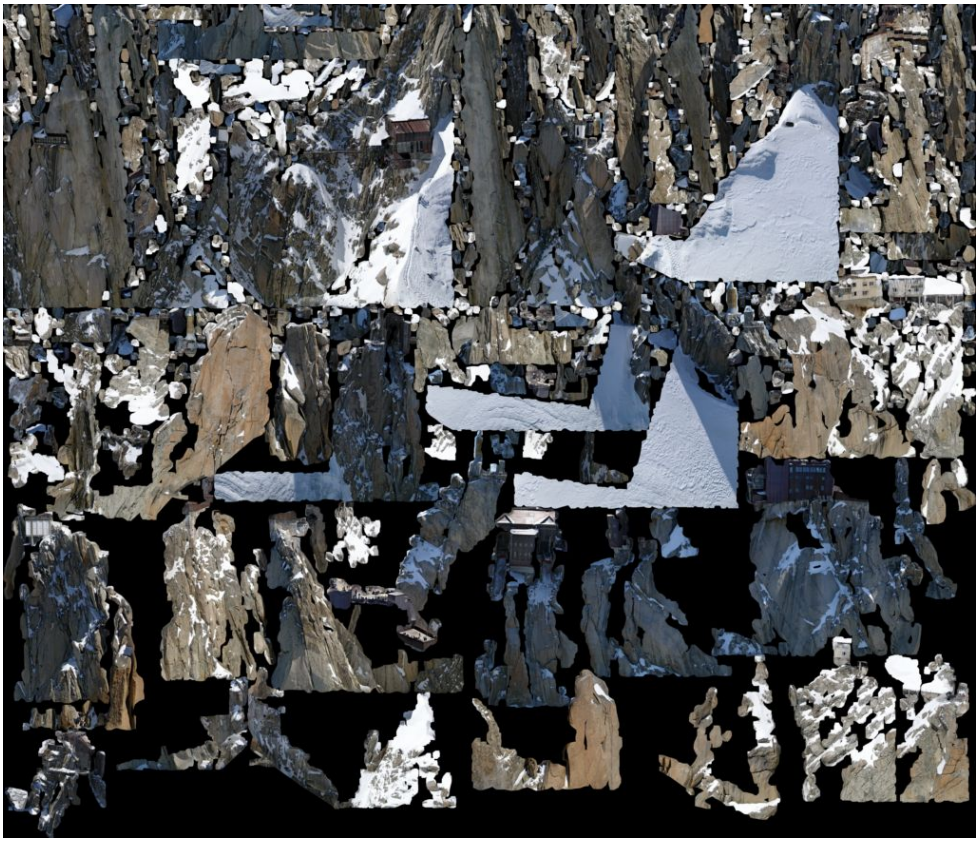


(a)



(b)

FIGURE 8.23 – Reconstruction 3D du château d'Ettlingen avec la partition des textures générées par : (a) l'approche naïve ; (b) l'approche par optimisation de partition.



(a)

(b)

FIGURE 8.24 – Atlas de textures 3D pour : (a) l'Aiguille du Midi ; (b) le château d'Ettlingen.

8.5 Conclusion

La méthode que nous avons proposé permet de créer des atlas de textures sans jointure visible pour des modèles 3D de scènes réelles reconstruits à partir de vues multiples.

Avant la construction de l'atlas à proprement parler, notre méthode repose sur une première étape d'optimisation de la partition de la texture. Bien que nous ayons mis en évidence un critère efficace et très simple à utiliser, le résultat de la minimisation par coupes minimales ne garantit pas l'invisibilité des jointures entre les différents fragments de la texture. C'est pourquoi une seconde étape s'avère nécessaire. La décomposition en bandes de fréquences est des plus intéressantes dans le cadre du mélange d'images puisqu'elle assure la continuité des couleurs sans pour autant créer d'artefacts de type fantôme ou de flou. Or, de par les imprécisions de la calibration ou les imperfections de la construction du modèle 3D, de tels artefacts sont très susceptibles de se produire avec des méthodes classiques telles que celle par dégradé de canal alpha. Bien sûr, pour pouvoir l'appliquer à notre cas, il nous a fallu étendre la méthode proposée dans [42] à des images de dimensions quelconques et en nombre supérieur à deux. Cela nous a ainsi permis, par le biais des fonctions de projection, d'obtenir une texture sans jointure visible.

Bien que grandement dépendants de la qualité de la calibration et de la construction du maillage faits en amonts, les résultats que nous avons obtenus avec notre méthode sur des exemples de reconstructions de monuments sont prometteurs.

Conclusion

Arrivés au terme de ce mémoire, nous procédons à la synthèse des travaux qui y ont été présentés afin de rappeler ici nos contributions ainsi que les perspectives qu'elles offrent.

Apports théoriques

La première partie de ce manuscrit fut consacrée aux recherches théoriques sur les différents paradigmes de segmentation de graphes.

Après avoir défini notre cadre de travail que sont les graphes et rappelé les terminologies dont nous avons eu besoin pour la suite de ce mémoire dans le chapitre 1, nous avons mis en lumière, dans le chapitre 2, les avantages et inconvénients qu'entraîne l'utilisation de l'un ou l'autre des différents types de segmentation de graphes : ensemble frontière de sommets, ensemble frontière d'arêtes et coupe. Il a résulté de cette étude que les coupes offrent plusieurs avantages non négligeables par rapport aux ensembles frontières de sommets ou d'arêtes.

Différents paradigmes de segmentation de graphes pondérés furent présentés dans les chapitres 3, 4 et 5 afin de permettre la mise en évidence de certains liens existant entre eux.

La comparaison, dans le chapitre 3, de différentes définitions de ligne de partage des eaux (LPE) parmi les plus usitées ou les plus récentes a nécessité un effort de généralisation, pour adapter certaines d'entre elles au cas d'un graphe à sommets ou à arêtes valués indépendamment, ou de redéfinition, pour déduire, de celles dont le résultat, dans sa définition originelle, est une application de poids sur le graphe, une segmentation de graphe qui soit comparable aux autres définitions de LPE. Il résulte de ces travaux que la définition de LPE d'arêtes sur graphes à arêtes valuées s'avère posséder bon nombre de propriétés intéressantes et que, de plus, la définition de celle-ci est en étroite relation avec plusieurs définitions des LPE sur graphes à sommets valués adaptées pour les graphes à arêtes valuées. Cette définition permet donc en quelque sorte d'"unifier" la plupart des définitions exposées dans ce manuscrit tout en regroupant l'ensemble des propriétés rencontrées. Il est à rajouter que la LPE d'arêtes est la seule définition à avoir des algorithmes s'exécutant en temps linéaire (dans le cas où elle est relative aux minima du graphe et non à un sous-graphe quelconque).

D'autres paradigmes de segmentation de graphes reposant sur des coupes induites par forêts couvrantes optimales relatives à sous-graphes furent rappelés, dans le chapitre 4, afin de faire l'objet de comparaisons avec la LPE d'arêtes précédemment retenue. Certains liens connus entre forêt couvrante de poids minimal (FCMin), forêt couvrante de poids maximal (FCMax) et forêt couvrante de chemins de moindre altitude (FCCMA) relatives à un sous-graphe furent donc rappelés afin de mieux appréhender la mise en évidence des nouveaux liens, respectivement d'implication et d'équivalence, entre coupes induites par forêts couvrantes, de poids minimal ou de chemins de moindre altitude, et LPE d'arêtes relatives à un sous-graphe quelconque. Les propriétés offertes par les FCMin et l'existence

d'algorithmes quasi-linéaires pour les déterminer nous permettent d'en conclure que la recherche d'une LPE d'arêtes par le biais d'une recherche de coupe induite par FCMin est à préconiser.

Les relations entre les paradigmes de segmentation de graphes précédents et la coupe minimale furent ensuite étudiés. La détermination de cette dernière se fait par le biais d'une recherche de flot maximal dans un ou plusieurs réseaux, qui sont des graphes orientés particuliers possédant en sommets de références une source et un puits. Il existe de nombreux algorithmes, de complexités différentes, permettant cette recherche. Ceux-ci peuvent être regroupés en deux catégories de recherche de flot maximal : par chemins améliorants ou par pré-flots. Les principes sur lesquels reposent ces deux catégories sont exposés dans le chapitre 5. Leur étude permit la mise en évidence que, pour certaines conditions sur la pondération des arêtes du graphe, une coupe minimale est une coupe induite par FCMax. Cette condition correspond à une certaine mise à la puissance n des poids du graphe. Il fut alors possible d'en déduire qu'une coupe par FCMax correspond à une minimisation globale au sens de la norme infinie et que cette dernière peut être considérée comme une heuristique gloutonne d'approximation d'une coupe minimale ayant l'avantage d'avoir des algorithmes quasi-linéaires et donnant un résultat d'autant plus proche que les écarts entre les poids des arêtes du graphe sont grands.

Ci-dessous se trouvent les relations, déjà connues et rappelées dans le chapitre 4, entre FCMin, FCMax et FCCMA relatives à un sous-graphe M sur le graphe G pondéré sur les arêtes par P ainsi que LPE d'arêtes relative à M sur G pondéré par P :

- Toute FCMin est une FCMax, et réciproquement, par l'application d'une fonction strictement décroissante sur P.
- Toute FCMin est une FCCMA et donc toute coupe par FCMin est une coupe par FCCMA.
- Si $M = Min(P)$, FCMin et FCCMA sont équivalentes et donc coupes par FCMin et FCCMA sont équivalentes.
- Si $M = Min(P)$, coupe par FCMin, coupe par FCCMA et LPE sont équivalentes.

En guise de récapitulatif de nos travaux théoriques, nous rajoutons ainsi aux relations sus-mentionnées les suivantes, nouvellement mises en évidence dans ce manuscrit :

- Toute coupe induite par FCMin est une LPE.
- Coupe induite par FCCMA et LPE sont équivalentes.
- Il existe un nombre réel $m \in \mathbb{R}^+$ tel que, pour tout $n \geq m$, une coupe minimale pour $P^{[n]}$ est une coupe par FCMax pour $P^{[n]}$ (et donc pour P également).

Ainsi, nous pouvons nous rendre compte que coupe par FCMin, coupe par FCMax, coupe par FCCMA, LPE d'arêtes et coupe minimale sont toutes liées par certaines conditions.

Apports appliqués

La seconde partie de ce mémoire de thèse fut consacrée à deux applications réalisées avec des méthodes de segmentation de graphes. Ces méthodes, reposant sur la recherche de coupes minimales appelées, respectivement, $\{\alpha, \beta\}$-*swaps* et α-*expansions*, sont exposées dans le chapitre 6. Ces méthodes reçurent quelques améliorations de notre cru visant à réduire le nombre de sommets et d'arcs dans les graphes de recherche des coupes minimales par rapport aux versions originelles, réduisant ainsi les temps de recherche. Elles permettent d'obtenir une minimisation d'énergie correspondant à un problème d'étiquetage sur un champ aléatoire de Markov. Le résultat obtenu n'est cependant, dans la plupart des cas, qu'une approximation du résultat optimal. Leurs conditions d'utili-

sation diffèrent sensiblement. Les $\{\alpha, \beta\}$-*swaps* sont plus générales, mais les α-*expansions* sont plus rapides. Ces dernières sont donc souvent préférées. Ce sont d'ailleurs celles que nous avons employées dans nos applications.

Notre première application, décrite dans le chapitre 7 et baptisée *renaissance d'image* concerna la complétion d'image : compléter les régions manquantes, occultées ou endommagées d'une image à partir de ses régions intactes de sorte à obtenir en résultat une image qui soit visuellement plausible. Ce problème fut reformulé sous la forme d'un problème d'étiquetage lié à une énergie à minimiser par la méthode des α-*expansions* consistant à copier les valeurs de pixels situées dans la partie intacte de l'image dans sa partie à reconstruire de sorte à préserver une continuité des couleurs avec la partie intacte. Cependant, devant le très grand nombre d'étiquettes impliquées dans une telle minimisation, il nous fallut trouver un moyen de faire une pré-sélection des étiquettes les plus probablement utilisées pour en limiter le nombre et ainsi obtenir un résultat en un temps acceptable. Cette pré-sélection fut mise en place au moyen de filtres à particules. Ceux-ci permettent de détecter rapidement des zones de la partie intacte de l'image ayant un contenu similaire à celui au bord de la partie à reconstruire et donc susceptibles d'y être copiés. Des résultats concluants, y compris pour la complétion d'images fortement texturées, furent obtenus.

Notre seconde application, décrite dans le chapitre 8, concerna la synthèse d'un atlas de textures sans jointure pour la reconstruction 3D de scènes réelles à partir de vues multiples. Créer la texture d'un maillage construit à partir de photographies prises sous différents points de vue consiste à déterminer l'image qui, pour chaque facette, sera projetée sur le maillage. Ce problème fut reformulé sous la forme d'un problème d'étiquetage lié à une énergie à minimiser par la méthode des α-*expansions* consistant à maximiser la résolution de la texture sur chaque facette tout en minimisant les dissimilarités entre deux facettes. Néanmoins, malgré une mise en oeuvre efficace de cette formulation, les résultats obtenus montraient de disgracieuses jointures causées par de trop grandes différences dans les conditions de luminosité entre les prises de vue de la scène. Pour y remédier nous mîmes donc en place un mélange de textures entre les facettes étiquetées différemment. Nous optâmes pour une méthode de mélange par décomposition en bandes de fréquences afin d'obtenir une transition douce des couleurs sans pour autant créer de flou ou d'artefacts de type fantômes (susceptibles de se produire quand les résultats de calibration et/ou de reconstruction ne sont pas parfaits). Nous dûmes donc généraliser la méthode originelle pour un nombre quelconque d'images de tailles elles aussi quelconques avant de pouvoir l'appliquer sur la texture par le biais des différentes fonctions de projections des images. Enfin la création d'atlas fut mise en place de sorte à regrouper les parties des vues utilisées dans notre texture dans une seule image de taille la plus petite possible, rendant ainsi l'exportation de nos résultats possible vers la majeure partie des programmes de visualisation d'objets 3D. Des résultats concluants furent obtenus pour la synthèse de textures de deux scènes extérieures reconstruites : l'Aiguille du Midi (Mont-Blanc, Chamonix, France) et le château d'Ettlingen (Ettlingen, Allemagne).

Perspectives

Les études théoriques entre les différents paradigmes de segmentation de graphes étudiés dans la première partie de ce manuscrit nous ont permis de mettre en relation les coupes induites par des forêts couvrantes optimales avec certaines définitions de LPE ainsi que les coupes minimales pour les forêts couvrantes de poids extremum. Si le premier lien nous permet de préconiser l'utilisation d'un algorithme de recherche de FCMin

pour la recherche d'une LPE relative, le second ne semble pas des plus applicables. En effet, d'après ce que nous avons montré, nous pouvons obtenir une coupe induite par FCMax à l'aide d'un algorithme de recherche de coupe minimale après avoir modifié la pondération du graphe en conséquence. Ceci n'a absolument aucun intérêt en soi puisqu'il existe des méthodes de recherche de FCMax en temps quasi-linéaire alors que la recherche d'une coupe minimale se fait en temps polynomial et, qui plus est, n'offre qu'un résultat approché si on considère une segmentation en plus de deux régions. Un lien "inverse" aurait donc été souhaitable... Malheureusement, à notre connaissance, un tel lien n'existe pas. Il serait néanmoins intéressant d'étudier certaines propriétés qui pourraient être liées à cette relation. En effet, utiliser la coupe induite par la FCMax comme initialisation d'une recherche de coupe minimale par *active cuts* (section 5.3.3.2) pourrait peut-être réduire les temps de calcul. De même, il n'est pas inenvisageable qu'un algorithme se basant sur la recherche dynamique d'une FCMax dans le graphe résiduel du réseau permettent également de réduire, peut-être pas la complexité de calcul, mais les temps de recherche de façon similaire à l'algorithme proposé par Yuri Boykov et Vladimir Kolmogorov [32, 33] (voir section 5.3.1.4). Tirer partie de ce lien, de façon indirecte, ne semble donc pas impossible même si nous n'avons pas eu le temps d'explorer davantage cette piste durant cette thèse.

La technique de *renaissance d'image* que nous proposons offre des résultats concluants, y compris pour la complétion d'images fortement texturées. Toutefois, sa trop grande dépendance à la phase de pré-sélection est un point faible. En effet, une première phase trop sélective et l'absence de certaines étiquettes pourrait s'avérer néfaste au résultat final. Une pré-sélection trop large et c'est le temps de calcul qui devient handicapant. De plus, l'aspect probabiliste et la détermination aléatoire de notre filtre à particules ne garantit pas nécessairement l'obtention systématique des meilleures étiquettes. Une phase de recherche concernant son remplacement par une autre méthode plus déterministe mais néanmoins rapide pourrait donc être envisagée.

La formulation de la complétion d'image sous forme d'un problème d'étiquetage lié à une minimisation d'énergie n'avait, à notre connaissance, jamais été faite avant nous. Certes, il n'est pas toujours possible de compléter une image par copie de régions de pixels en provenance de la partie intacte dans la partie à reconstruire, mais, sous cette hypothèse, ce type de formulation s'avère des plus prometteurs. Elle pourrait sans doute être étendu à la complétion de vidéos ou même à la complétion de textures d'un modèle 3D. Néanmoins, un intérêt tout particulier devra être porté à la phase de pré-sélection des étiquettes d'autant plus que la dimension supplémentaire liée à ces problèmes augmente de façon considérable l'espace de recherche.

Les méthodes automatisées de reconstruction 3D d'objets ou de scènes réels à partir de photo prises sous différents points de vue deviennent de plus en plus efficaces et sont en passe de se démocratiser tant la demande est grande dans de nombreux domaines nécessitant la reproduction d'éléments réels dans un monde virtuel pour permettre leur manipulation : effets spéciaux de cinéma, tourisme, vente en ligne, jeux vidéo,...

Des améliorations sont régulièrement apportées aux méthodes de calibration et de construction du maillage. En effet, si, il y a seulement quelques années, on arrivait déjà à reconstruire sans trop de difficulté un objet de forme convexe ou s'en approchant avec de bonnes conditions de prise de vue, il n'en était pas de même pour des objets plus complexes ou lorsque l'arrière-plan n'était pas d'une couleur unie. Les techniques actuelles que nous avons employées pour la construction des maillages utilisés dans nos résultats font parties des plus efficaces existant actuellement. Néanmoins, certains perfectionnements sont encore à apporter pour augmenter la fidélité de la reconstruction.

La technique de création de texture pour un tel maillage devient donc un enjeu d'im-

portance puisque, bien que bonne, la reconstruction n'est pas encore parfaite. La détermination de la texture à partir des images ayant servi à la construction du maillage pâtit de ces imperfections et risque ainsi de créer des artefacts indésirables dus aux erreurs de construction préalables. Notre méthode y remédie par la phase de mélange de textures. Toutefois, même s'il est juste d'espérer que les résultats de reconstruction continueront de s'améliorer dans l'avenir, nous ne pourrons pas pour autant nous contenter de la phase d'étiquetage puisque les différences de conditions de luminosité demeureront toujours un problème à prendre en compte.

Bibliographie

[1] Wilhelm Ackermann : Zum hilbertschen aufbau der reellen zahlen. *Mathematische Annalen*, 99:118–133, 1928. (Cité page 124.)

[2] David Adalsteinsson et James A. Sethian : A fast level set method for propagating interfaces. *Journal of Computational Physics*, 118(2):269–277, 1995. (Cité page 209.)

[3] Ravindra K. Ahuja, Thomas L. Magnanti et James B. Orlin : *Network flows : theory, algorithms, and applications*. Prentice Hall, 1993. (Cité page 137.)

[4] Ravindra K. Ahuja et James B. Orlin : A fast and simple algorithm for the maximum flow problem. Rapport technique 1905-87, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1987. (Cité page 153.)

[5] Ravindra K. Ahuja, James B. Orlin et Robert Endre Tarjan : Improved time bounds for the maximum flow problem. Rapport technique CS-TR-118-87, Princeton Univertsity, Depatment of Computer Science, 1987. (Cité page 153.)

[6] Ravindra K. Ahuja, James B. Orlin et Robert Endre Tarjan : Improved time bounds for the maximum flow problem. *Society for Industrial and Applied Mathematics Journal on Computing*, 18(5):939–954, 1989. (Cité page 153.)

[7] Cédric Allène, Jean-Yves Audibert, Michel Couprie, Jean Cousty et Renaud Keriven : Some links between min-cuts, optimal spanning forests and watersheds. Rapport technique IGM2007-06, Université Paris-Est, France, July 2007. (Cité pages vi, 47, 128 et 161.)

[8] Cédric Allène, Jean-Yves Audibert, Michel Couprie, Jean Cousty et Renaud Keriven : Some links between min-cuts, optimal spanning forests and watersheds. *In Proceedings of the 8th International Symposium on Mathematical Morphology*, pages 253–264, Rio de Janeiro, Brazil, October 2007. INPE. (Cité pages vi, 47 et 161.)

[9] Cédric Allène et Nikos Paragios : Image renaissance using discrete optimization and the alpha-expansion algorithm. Rapport technique 05-11, ENPC-CERTIS, France, September 2005. (Cité page 215.)

[10] Cédric Allène et Nikos Paragios : Image renaissance using discrete optimization. *In Proceedings of the 18th International Conference on Pattern Recognition*, Hong-Kong, China, August 2006. (Cité page 215.)

[11] Cédric Allène, Jean-Philippe Pons et Renaud Keriven : Seamless image-based texture atlases using multi-band blending. *In Proceedings of the 19th International Conference on Pattern Recognition*, Tampa, December 2008. (Cité page 238.)

[12] Ben Appleton et Hugues Talbot : Globally minimal surfaces by continuous maximal flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):106–118, 2006. (Cité page 211.)

[13] Sanjeev M. Arulampalam, Simon Maskell, Neil J. Gordon et Tim C. Clapp : A tutorial on particle filters for on-line non-linear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2002. (Cité page 220.)

[14] Romaric Audigier et Roberto de Alencar Lotufo : Tie-zone watershed, bottlenecks, and segmentation robustness analysis. *In Proceedings of the 18th Brazilian Symposium*

on Computer Graphics and Image Processing, pages 55–62. IEEE Computer Society, 2005. (Cité page 111.)

[15] Romaric Audigier et Roberto de Alencar Lotufo : Duality between the watershed by image foresting transform and the fuzzy connectedness segmentation approaches. *In Proceedings of the 19th Brazilian Symposium on Computer Graphics and Image Processing*, pages 53–60, 2006. (Cité page 125.)

[16] Romaric Audigier et Roberto de Alencar Lotufo : Watershed by image foresting transform, tie-zone, and theoretical relationships with other watershed definitions. *In Proceedings of the 8th International Symposium on Mathematical Morphology*, pages 277–288, 2007. (Cité page 111.)

[17] Romaric Audigier, Roberto de Alencar Lotufo et Michel Couprie : The tie-zone watershed : definition, algorithm and application. *In Proceedings of the 12th IEEE International Conference on Image Processing*, September 2005. (Cité page 111.)

[18] Paul Gustav Heinrich Bachmann : *Die analytische zahlentheorie*. Teubner, 1894. (Cité page 21.)

[19] Adam Baumberg : Blending images for texturing 3D models. *In Proceedings of the 13th British Machine Vision Conference*, pages 404–413, 2002. (Cité pages 236 et 237.)

[20] Richard Beare : Efficient implementation of the locally constrained watershed transform and seeded region growing. *In Proceedings of the 7th International Symposium on Mathematical Morphology*, pages 217–226, 2005. (Cité page 116.)

[21] Lowell W. Beineke : *On derived graphs and digraphs*, pages 17–33. Teubner-Verlag, 1968. (Cité page 31.)

[22] Claude Berge : *Theory of graphs*. Methuen and Co., 1962. (Cité page 3.)

[23] Fausto Bernardini, Ioana M. Martin et Holly Rushmeier : High-quality texture reconstruction from multiple scans. *IEEE Transactions on Visualization and Computer Graphics*, 7(4):318–332, 2001. (Cité pages 236 et 237.)

[24] Marcelo Bertalmío, Vicent Caselles, Gloria Haro et Guillermo Sapiro : Pde-based image and surface inpainting. *In Handbook of mathematical models in computer vision*, chapitre 3, pages 33–61. Springer, 2005. (Cité page 214.)

[25] Marcelo Bertalmío, Guillermo Sapiro, Vicent Caselles et Coloma Ballester : Image inpainting. *In Proceedings of the 27th International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 417–424, New York, NY, USA, 2000. Association for Computing Machinery. (Cité pages xxvi, 213, 214, 216, 223, 225, 226, 227, 228 et 230.)

[26] Gilles Bertrand : On topological watersheds. *Journal of Mathematical Imaging and Vision*, 22(2-3):217–230, May 2005. (Cité pages 47, 88 et 89.)

[27] Julian E. Besag : Spatial interaction and the statistical analysis of lattice systems. *Journal of Royal Statistical Society*, (36):192–236, 1974. (Cité pages 178 et 179.)

[28] Julian E. Besag : On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society*, 48:259–302, 1986. (Cité page 209.)

[29] Serge Beucher : Watershed, hierarchical segmentation and waterfall algorithm. *In Proceedings of the Mathematical Morphology and its Applications to Image Processing*, pages 69–76, September 1994. (Cité page 75.)

[30] Serge Beucher et Christian Lantuéjoul : Use of watersheds in contour detection. *In Proceedings of the International Workshop on Image Processing Real-Time Edge and Motion Detection/Estimation*, 1979. (Cité pages 75 et 77.)

[31] Serge Beucher et Fernand Meyer : The morphological approach to segmentation : the watershed transformation. *In* E. R. Dougherty, éditeur : *Mathematical Morphology in Image Processing*, chapitre 12, pages 433–481. Marcel Dekker, New York, 1993. (Cité pages 75 et 82.)

[32] Yuri Boykov et Vladimir Kolmogorov : An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *In Proceedings of the 2nd International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 359–374, London, UK, 2001. Springer-Verlag. (Cité pages 150, 182, 183 et 272.)

[33] Yuri Boykov et Vladimir Kolmogorov : An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004. (Cité pages 150, 182, 183 et 272.)

[34] Yuri Boykov et Olga Veksler : Graph cuts in vision and graphics : theories and applications. *In Handbook of mathematical models in computer vision*, chapitre 5, pages 79–96. Springer, 2005. (Cité page 183.)

[35] Yuri Boykov, Olga Veksler et Ramin Zabih : Markov random fields with efficient approximations. *In Proceedings of the 1998 IEEE Computer Society conference on Computer Vision and Pattern Recognition*, pages 648–655. IEEE Computer Society, 1998. (Cité pages 175, 182, 191, 192, 198, 206 et 208.)

[36] Yuri Boykov, Olga Veksler et Ramin Zabih : Fast approximate energy minimization via graph cuts. *In Proceedings of the 7th IEEE International Conference on Computer Vision*, pages 377–384, Washington, DC, USA, 1999. IEEE Computer Society. (Cité pages 175, 182, 191, 192, 194 et 198.)

[37] 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. (Cité pages 175, 182 et 191.)

[38] 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. (Cité pages 175, 182, 191, 192, 194, 198, 202 et 203.)

[39] Matthew Brown et David G. Lowe : Recognising panoramas. *In Proceedings of the 9th IEEE International Conference on Computer Vision*, volume 2, pages 1218–1225, Washington, DC, USA, 2003. IEEE Computer Society. (Cité page 245.)

[40] Thomas Brox, Andrés Bruhn et Joachim Weickert : Variational motion segmentation with level sets. *In Proceedings of the 9th European Conference on Computer Vision*, volume 3951, pages 471–483, Graz, Austria, 2006. Springer-Verlag. (Cité page 209.)

[41] Peter J. Burt et Edward H. Adelson : The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, COM-31(4):532–540, 1983. (Cité pages 245, 247 et 248.)

[42] Peter J. Burt et Edward H. Adelson : A multiresolution spline with application to image mosaics. *In Proceedings of the 10th International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, volume 2, pages 217–236, New York, NY, USA, 1983. Association for Computing Machinery. (Cité pages 237, 245, 255 et 268.)

[43] Vicent Caselles, Guillermo Sapiro, Coloma Ballester, Marcelo Bertalmío et Joan Verdera : Filling-in by joint interpolation of vector fields and grey levels. *IEEE Transactions on Image Processing*, 10:1200–1211, 2001. (Cité page 214.)

[44] A. Cayley : On contour and slope lines. *The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science*, 18(120):264–268, 1859. (Cité page 75.)

[45] Bernard Chalmond : *Éléments de modélisation pour l'analyse d'images*, volume 33 de *Mathématiques et applications*. Springer-Verlag, 2000. (Cité page 177.)

[46] Tony F. Chan et Jianhong Shen : Mathematical models of local non-texture inpaintings. *Society for Industrial and Applied Mathematics Journal on Applied Mathematics*, 62:1019–1043, 2001. (Cité page 214.)

[47] Tony F. Chan et Jianhong Shen : Non-texture inpaintings by curvature-driven diffusions (CDD). *Journal of Visual Communication and Image Representation*, 12(4):436–449, 2001. (Cité page 214.)

[48] Bernard Chazelle : A minimum spanning tree algorithm with inverse-Ackermann type complexity. *Journal of the Association for Computing Machinery*, 47:1028–1047, 2000. (Cité pages 124, 132 et 135.)

[49] Laurent D. Cohen : Minimal paths and fast marching methods for image analysis. *In Handbook of mathematical models in computer vision*, chapitre 6, pages 97–111. Springer, 2005. (Cité page 209.)

[50] Laurent D. Cohen et Ron Kimmel : Fast marching the global minimum of active contours. *In Proceedings of the 3rd IEEE International Conference on Image Processing*, pages 473–476, Lausanne, Suisse, September 1996. (Cité page 209.)

[51] Laurent D. Cohen et Ron Kimmel : Global minimum for active contour models : a minimal path approach. *International Journal of Computer Vision*, 24(1):57–78, 1997. (Cité page 209.)

[52] William J. Cook, William H. Cunningham, William R. Pulleyblank et Alexander Schrijver : *Combinatorial optimization*. John Wiley & Sons, Inc., New York, NY, USA, 1998. (Cité page 146.)

[53] Thomas H. Cormen, Charles E. Leiserson et Ronald L. Rivest : *Introduction to algorithms, second edition*. MIT Press, 2001. (Cité pages 22, 124 et 150.)

[54] Michel Couprie et Gilles Bertrand : Topological grayscale watershed transformation. *In SPIE Vision Geometry VI Proceedings*, volume 3168, pages 136–146, 1997. (Cité pages 88 et 89.)

[55] Michel Couprie, Laurent Najman et Gilles Bertrand : Quasi-linear algorithms for the topological watershed. *Journal of Mathematical Imaging and Vision*, 22(2–3):231–249, May 2005. Special issue on Mathematical Morphology. (Cité pages 88 et 89.)

[56] Jean Cousty, Gilles Bertrand, Laurent Najman et Michel Couprie : Fusion graphs : merging properties and watersheds. Rapport technique IGM2005-04, Université Paris-Est, France, 2005. (Cité pages 58, 59, 60, 61, 64 et 67.)

[57] Jean Cousty, Gilles Bertrand, Laurent Najman et Michel Couprie : Watershed cuts. *In Proceedings of the 8th International Symposium on Mathematical Morphology*, pages 301–312, 2007. (Cité pages 99, 100 et 105.)

[58] Jean Cousty, Gilles Bertrand, Laurent Najman et Michel Couprie : Watersheds, minimum spanning forests and the drop of water principle. Rapport technique IGM2007-01, Université Paris-Est, France, 2007. (Cité pages 99, 100, 105, 114, 125 et 128.)

[59] Jean Cousty, Michel Couprie, Laurent Najman et Gilles Bertrand : Grayscale watersheds on perfect fusion graphs. *In Proceedings of the 11th International Workshop on Combinatorial Image Analysis*, volume 4040 de *Lecture Notes in Computer Science*, pages 60–73. Springer, 2006. (Cité pages 92, 97 et 113.)

[60] Jean Cousty, Michel Couprie, Laurent Najman et Gilles Bertrand : Weighted fusion graphs : merging properties and watersheds. Rapport technique IGM2007-09, Université Paris-Est, France, 2007. (Cité pages 92, 97 et 113.)

[61] Antonio Criminisi, Patrick Pérez et Kentaro Toyama : Object removal by exemplar-based inpainting. *In Proceedings of the 2003 IEEE Computer Society conference on Computer Vision and Pattern Recognition*, pages 721–728, 2003. (Cité page 214.)

[62] Gruia Călinescu, Howard Karloff et Yuval Rabani : An improved approximation algorithm for multiway cut. *Journal of Computer and System Sciences*, 60(3):564–574, 2000. (Cité page 159.)

[63] William H. Cunningham : The optimal multiterminal cut problem. *In Reliability of computer and communication networks*, volume 5 de *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 105–120. American Mathematical Society, 1991. (Cité page 159.)

[64] Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour et Mihalis Yannakakis : The complexity of multiway cuts. *In Unpublished extended abstract*, 1983. (Cité page 159.)

[65] Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour et Mihalis Yannakakis : The complexity of multiway cuts. *In Proceedings of the 24th Annual Association for Computing Machinery Symposium on the Theory of Computing*, pages 241–251, New York, NY, USA, 1992. Association for Computing Machinery. (Cité page 159.)

[66] Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour et Mihalis Yannakakis : The complexity of multiterminal cuts. *Society for Industrial and Applied Mathematics Journal on Computing*, 23:864–894, 1994. (Cité pages xxv, 138, 158 et 159.)

[67] 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. (Cité page 144.)

[68] Paul E. Debevec, Camillo J. Taylor et Jitendra Malik : Modeling and rendering architecture from photographs : a hybrid geometry and image-based approach. *In Proceedings of the 23rd International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 11–20, New York, NY, USA, 1996. Association for Computing Machinery. (Cité pages 236 et 259.)

[69] Daniele Giorgio Degiorgi et Klaus Simon : A dynamic algorithm for line graph recognition. *In Proceedings of the 21st International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 37–48, London, UK, 1995. Springer-Verlag. (Cité page 31.)

[70] Reinhard Diestel : *Graph theory*. Graduate Texts in Mathematics. Springer, 1997. (Cité page 144.)

[71] H. Digabel et Christian Lantuéjoul : Iterative algorithm. *In Proceedings of the 2nd European Symposium on Quantitive Analysis of Microstructures in Material Science, Biology and Medicine*, October 1977. (Cité page 75.)

[72] E. A. Dinic : Algorithm for solution of a problem of maximum flow in networks with power estimation. *Soviet Mathematics Doklady*, 11:1277–1280, 1970. (Cité page 150.)

[73] Arnaud Doucet : On sequential simulation-based methods for Bayesian filtering. Rapport technique CUED/F-INFENG/TR. 310, Cambridge University Department of Engineering, 1998. (Cité page 220.)

[74] Arnaud Doucet, Nando de Freitas et Neil J. Gordon : *Sequential Monte Carlo methods in practice*. Springer-Verlag, New-York, 2001. (Cité page 220.)

[75] Arnaud Doucet, Simon Godsill et Christophe Andrieu : On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000. (Cité page 220.)

[76] Romain Dupont, Olivier Juan et Renaud Keriven : Robust segmentation of hidden layers in video sequences. Rapport technique 06-21, ENPC-CERTIS, France, January 2006. (Cité page 231.)

[77] Romain Dupont, Olivier Juan et Renaud Keriven : Robust segmentation of hidden layers in video sequences. *In Proceedings of the 18th International Conference on Pattern Recognition*, Hong-Kong, China, August 2006. (Cité page 231.)

[78] 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, April 1972. (Cité page 150.)

[79] Alexei A. Efros et William T. Freeman : Image quilting for texture synthesis and transfer. *In Proceedings of the 28th International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 341–346, New York, NY, USA, 2001. Association for Computing Machinery. (Cité page 214.)

[80] Alexei A. Efros et Thomas K. Leung : Texture synthesis by non-parametric sampling. *In Proceedings of the 7th IEEE International Conference on Computer Vision*, pages 1033–1038, 1999. (Cité page 214.)

[81] Selim Esedoglu et Jianhong Shen : Digital inpainting based on the Mumford-Shah-Euler image model. *European Journal of Applied Mathematics*, 13:353–370, 2002. (Cité page 214.)

[82] Leonhard Euler : Solutio problematis ad geometriam situs pertinentis. *Commentarii Academiae Scientiarum Imperialis Petropolitanae*, 8:128–140, 1736. (Cité pages 3 et 4.)

[83] Alexandre X. Falcão, Jorge Stolfi et Roberto de Alencar Lotufo : The image foresting transform : theory, algorithm and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:19–29, 2004. (Cité pages 85 et 125.)

[84] Pedro F. Felzenszwalb et Daniel P. Huttenlocher : Efficient belief propagation for early vision. *In Proceedings of the 2004 IEEE Computer Society conference on Computer Vision and Pattern Recognition*, pages 261–268. IEEE Computer Society, 2004. (Cité page 210.)

[85] Lester Randolph Sr Ford et Delbert Ray Fulkerson : Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956. (Cité pages 137, 146 et 148.)

[86] Lester Randolph Sr Ford et Delbert Ray Fulkerson : *Flows in networks*. Princeton University Press, 1962. (Cité pages 146, 148, 154 et 159.)

[87] Daniel Freedman et Petros Drineas : Energy minimization via graph cuts : settling what is possible. *In Proceedings of the 2005 IEEE Computer Society conference on Computer Vision and Pattern Recognition*, pages 939–946, Washington, DC, USA, 2005. IEEE Computer Society. (Cité page 182.)

[88] Naveen Garg, Vijay V. Vazirani et Mihalis Yannakakis : Approximate max-flow min-(multi)cut theorems and their applications. *Society for Industrial and Applied Mathematics Journal on Computing*, 25:698–707, 1996. (Cité page 158.)

[89] Donald Geman, Stuart Geman, Christine Graffigne et Pinliang Dong : Boundary detection by constrained optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):609–628, 1990. (Cité page 181.)

[90] Stuart Geman et Donald Geman : Stochastic relaxation, Gibbs distributions and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984. (Cité pages 178, 179 et 209.)

[91] Andrew V. Goldberg, Eva Tardos et Robert Endre Tarjan : Network flow algorithm,. *Paths, flows and VLSI-design*, pages 101–164, 1998. (Cité page 146.)

[92] Andrew V. Goldberg et Robert Endre Tarjan : A new approach to the maximum flow problem. *In Proceedings of the 18th annual Association for Computing Machinery Symposium on Theory of Computing*, pages 136–146, New York, NY, USA, 1986. Association for Computing Machinery. (Cité pages 153 et 211.)

[93] Andrew V. Goldberg et Robert Endre Tarjan : Finding minimum-cost circulations by successive approximations. Rapport technique MIT/LCS/TM-333, Massachusetts Institute of Technology, Cambridge, 1987. (Cité pages 153 et 211.)

[94] Andrew V. Goldberg et Robert Endre Tarjan : A new approach to the maximum-flow problem. *Journal of the Association for Computing Machinery*, 35(4):921–940, 1988. (Cité pages 153 et 211.)

[95] Olivier Goldschmidt et Dorit S. Hochbaum : Polynomial algorithm for the k-cut problem. *In Proceedings of the 29th annual Symposium on Foundations of Computer Science*, pages 444–451, Washington, DC, USA, 1988. IEEE Computer Society. (Cité page 158.)

[96] Olivier Goldschmidt et Dorit S. Hochbaum : A polynomial algorithm for the k-cut problem for fixed k. *Mathematics of Operations Research*, 19:24–37, 1994. (Cité page 158.)

[97] Olivier Goldschmidt et David B. Shmoys : An $O(|V|^2)$ algorithm for the planar 3-cut problem. *Journal on Algebraic and Discrete Methods*, 6(4):707–712, 1985. (Cité page 158.)

[98] Neil J. Gordon, David J. Salmond et Adrian F. M. Smith : Novel approach to nonlinear/non-gaussian bayesian state estimation. *In IEE Proceedings F on Radar and Signal Processing*, volume 140, pages 107–113, 1993. (Cité page 220.)

[99] Nuno R. E. Gracias, Art C. R. Gleason, Shahriar Negahdaripour et Mohammad H. Mahoor : Fast image blending using watersheds and graph cuts. *In Proceedings of the 17th British Machine Vision Conference*, volume II, pages 469–478, 2006. (Cité page 245.)

[100] Leo Grady : Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1768–1783, 2006. (Cité page 210.)

[101] Ron Lewis Graham et Pavol Hell : On the history of the minimum spanning tree problem. *Annals of the History of Computing*, pages 43–57, 1985. (Cité page 124.)

[102] D. M. Greig, B. T. Porteous et A. H. Seheult : Discussion of : On the statistical analysis of dirty pictures (by j. e. besag.). *Journal of the Royal Statistical Society*, 48:282–284, 1986. (Cité page 209.)

[103] 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. (Cité page 182.)

[104] David J. Heeger et James R. Bergen : Pyramid-based texture analysis/synthesis. *In Proceedings of the 22nd International Computer on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 229–238, New York, NY, USA, 1995. Association for Computing Machinery. (Cité page 214.)

[105] T. C. Hu : *Integer programming and network flows*. Addison-Wesley, 1969. (Cité page 158.)

[106] Homan Igehy et Lucas Pereira : Image replacement through texture synthesis. *In Proceedings of the 4th IEEE International Conference on Image Processing*, pages 186–189, Washington, DC, USA, 1997. IEEE Computer Society. (Cité page 214.)

[107] Hiroshi Ishikawa : Exact optimization for Markov random fields with convex priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:1333–1336, 2003. (Cité page 206.)

[108] Hiroshi Ishikawa et Davi Geiger : Occlusions, discontinuities, and epipolar lines in stereo. *In Proceedings of the 5th European Conference on Computer Vision*, pages 232–248, London, UK, 1998. Springer-Verlag. (Cité page 206.)

[109] Zsolt Jankó : *Photorealistic 3D models of real-world objects*. Thèse de doctorat, Eotvos Lorand University, Hungary, 2007. (Cité page 236.)

[110] Zsolt Jankó et Dmitry Chetverikov : Photo-consistency based registration of an uncalibrated image pair to a 3D surface model using genetic algorithm. *In Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization and Transmission*, pages 616–622, Washington, DC, USA, 2004. IEEE Computer Society. (Cité page 237.)

[111] C. Jordan : Nouvelles observations sur les lignes de faîte et de thalweg. *Comptes rendus des séances de l'académie des sciences*, 75:1023–1025, 1872. (Cité page 75.)

[112] Olivier Juan et Yuri Boykov : Active cuts for real-time graph partitioning in vision. Rapport technique 655, University of Western Ontario, November 2005. (Cité page 154.)

[113] Olivier Juan et Yuri Boykov : Active graph cuts. *In Proceedings of the 2006 IEEE Computer Society conference on Computer Vision and Pattern Recognition*, volume 1, pages 1023–1029, Los Alamitos, CA, USA, June 2006. IEEE Computer Society. (Cité page 154.)

[114] Gaetano Kanizsa : *Gramática de la visión*. Paidos, 1986. (Cité page 214.)

[115] Richard Manning Karp : Reducibility among combinatorial problems. *In Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972. (Cité pages 24 et 28.)

[116] Alexander V. Karzanov : Determining the maximal flow in network by the method of preflows. *Soviet Mathematics Doklady*, 15:434–437, 1974. (Cité page 153.)

[117] S. Kirkpatrick, C. D. Gelatt et M. P. Vecchi : Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983. (Cité page 208.)

[118] Pushmeet Kohli et Philip H. S. Torr : Effciently solving dynamic markov random fields using graph cuts. *Proceedings of the 10th IEEE International Conference on Computer Vision*, 2:922–929, 2005. (Cité page 153.)

[119] Pushmeet Kohli et Philip H. S. Torr : Dynamic graph cuts for efficient inference in markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2079–2088, 2007. (Cité page 153.)

[120] Anil C. Kokaram : On missing data treatment for degraded video and film archives : a survey and a new Bayesian approach. *IEEE Transactions on Image Processing*, 13:397–415, 2004. (Cité page 231.)

[121] Vladimir Kolmogorov : Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10): 1568–1583, 2006. (Cité page 210.)

[122] Vladimir Kolmogorov et Carsten Rother : Comparison of energy minimization algorithms for highly connected graphs. Rapport technique MSR-TR-2006-19, Microsoft Research, 2006. (Cité page 210.)

[123] 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. (Cité pages 182, 185, 186, 195 et 203.)

[124] Nikos Komodakis et Georgios Tziritas : Image completion using global optimization. *In Proceedings of the 2006 IEEE Computer Society conference on Computer Vision and Pattern Recognition*, pages 442–452, Washington, DC, USA, 2006. IEEE Computer Society. (Cité page 231.)

[125] Nikos Komodakis et Georgios Tziritas : Approximate labeling via graph cuts based on linear programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8):1436–1453, 2007. (Cité page 211.)

[126] Nikos Komodakis, Georgios Tziritas et Nikos Paragios : Fast, approximately optimal solutions for single and dynamic mrfs. *In Proceedings of the 11th IEEE International Conference on Computer Vision*, pages 1–8. IEEE Computer Society, 2007. (Cité page 211.)

[127] Nikos Komodakis, Georgios Tziritas et Nikos Paragios : Performance vs computational efficiency for optimizing single and dynamic mrfs : Setting the state of the art with primal-dual strategies. *Computer Vision and Image Understanding*, 112(1):14–29, 2008. (Cité page 211.)

[128] Bernhard Korte et Jens Vygen : *Combinatorial optimization : theory and algorithms*. Algorithms and Combinatorics. Springer, 2000. (Cité pages 123, 124 et 134.)

[129] Joseph Bernard Jr Kruskal : On the shortest spanning tree of a graph and the traveling salesman problem. *In Proceedings of the American Mathematical Society*, volume 7, pages 48–50, 1956. (Cité page 124.)

[130] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk et Aaron Bobick : Graphcut textures : image and video synthesis using graph cuts. *In Proceedings of the 30th International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, volume 22, pages 277–286, New York, NY, USA, 2003. Association for Computing Machinery. (Cité pages 214 et 218.)

[131] Patrick Labatut, Jean-Philippe Pons et Renaud Keriven : Efficient multi-view reconstruction of large-scale scenes using interest points, Delaunay triangulation and graph cuts. *In Proceedings of the 11th IEEE International Conference on Computer Vision*. IEEE Computer Society, 2007. (Cité page 238.)

[132] Edmund Georg Hermann Landau : *Handbuch der lehre von der verteilung der primzahlen*. Teubner, 1909. (Cité page 21.)

[133] 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. (Cité page 31.)

[134] Victor S. Lempitsky et Denis V. Ivanov : Seamless mosaicing of image-based texture maps. *In Proceedings of the 2007 IEEE Computer Society conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2007. (Cité pages 236, 237, 243 et 244.)

[135] Victor S. Lempitsky, Carsten Rother et Andrew Blake : Logcut - Efficient graph cut optimization for Markov random fields. *In Proceedings of the 11th IEEE International Conference on Computer Vision*. IEEE Computer Society, 2007. (Cité page 211.)

[136] Hendrik P. A. Lensch, Wolfgang Heidrich et Hans-Peter Seidel : A silhouette-based algorithm for texture registration and stitching. *Graphical Models*, 63(4):245–262, 2001. (Cité pages 236 et 237.)

[137] Bruno Lévy, Sylvain Petitjean, Nicolas Ray et Jérome Maillot : Least squares conformal maps for automatic texture atlas generation. *In Proceedings of the 29th International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 362–371, New York, NY, USA, 2002. Association for Computing Machinery. (Cité page 237.)

[138] Shengping Ziqing Li : *Markov random field modeling in computer vision*. Springer-Verlag, London, UK, 1995. (Cité pages 177 et 181.)

[139] Wen-Chieh Lin, James H. Hays, Chenyu Wu, Vivek Kwatra et Yanxi Liu : A comparison study of four texture synthesis algorithms on regular and near-regular textures. Rapport technique CMU-RI-TR-04-01, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 2004. (Cité page 213.)

[140] Herve Lombaert, Yiyong Sun, Leo Grady et Chenyang Xu : A multilevel banded graph cuts method for fast image segmentation. *In Proceedings of the 10th IEEE International Conference on Computer Vision*, pages 259–265, Washington, DC, USA, 2005. IEEE Computer Society. (Cité page 154.)

[141] David Marr : *Vision*. W. H. Freeman & Company, New-York, 1982. (Cité page 2.)

[142] Simon Masnou et Jean-Michel Morel : Level lines based disocclusion. *In Proceedings of the 5th IEEE International Conference on Image Processing*, pages 259–263, Chicago, IL, 1998. (Cité page 214.)

[143] James Clerk Maxwell : On hills and dales. *Philosophical Magazine*, 4/40(269):421–427, 1870. (Cité page 75.)

[144] Fernand Meyer : Un algorithme optimal de ligne de partage des eaux. *In Proceedings of the 8th conference on Reconnaissance des Formes et Intelligence Artificielle*, volume 2, pages 847–857, Lyon-Villeurbanne, 1991. (Cité pages 75, 77, 79, 82 et 124.)

[145] Fernand Meyer : Minimum spanning forests for morphological segmentation. *In Proceedings of the 2nd International Conference on Mathematical Morphology and its Applications to Image Processing*, pages 77–84, 1994. (Cité pages 105 et 132.)

[146] Fernand Meyer : Topographic distance and watershed lines. *Signal Processing*, 38: 113–125, 1994. (Cité pages 75 et 85.)

[147] Fernand Meyer : Levelings : theory and practice. *In Handbook of mathematical models in computer vision*, chapitre 4, pages 65–78. Springer, 2005. (Cité page 105.)

[148] Fernand Meyer et Serge Beucher : Morphological segmentation. *Journal of Visual Communication and Image Representation*, 1(1):21–46, 1990. (Cité pages 75 et 105.)

[149] Fernand Meyer et Laurent Najman : *Morphologie mathématique 1 : approches déterministes*, chapitre Segmentation, arbre de poids minimum et hiérarchies, pages 201–232. Traité IC2, série signal et image. Hermès Science Publications, 2008. (Cité page 132.)

[150] Fernand Meyer et Corinne Vachier : Image segmentation based on viscous flooding simulation. *In Proceedings of the 6th International Symposium on Mathematical Morphology*, pages 69–78. INPE, 2002. (Cité page 116.)

[151] Hiroshi Nagamochi et Toshihide Ibaraki : Computing edge-connectivity in multigraphs and capacitated graphs. *Journal on Algebraic and Discrete Methods*, 5(1):54–66, 1992. (Cité page 158.)

[152] Laurent Najman et Michel Couprie : Building the component tree in quasi-linear time. *Image Processing, IEEE Transactions on*, 15(11):3531–3539, 2006. (Cité page 89.)

[153] Laurent Najman, Michel Couprie et Gilles Bertrand : Watersheds, mosaics and the emergence paradigm. *Discrete Applied Mathematics*, 147(2-3):301–324, April 2005. Special issue on DGCI. (Cité pages 88 et 89.)

[154] Laurent Najman et Michel Schmitt : Watershed of a continuous function. *Signal Processing*, 38:99–112, 1994. (Cité page 85.)

[155] Laurent Najman et Michel Schmitt : Watershed of a continuous function. *Signal Processing*, 38(1):99–112, 1994. (Cité page 116.)

[156] Laurent Najman et Michel Schmitt : Geodesic saliency of watershed contours and hierarchical segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(12):1163–1173, December 1996. (Cité page 132.)

[157] Jaroslav Nešetřil, Eva Milková et Helena Nešetřilova : Otakar Boruvka on minimum spanning tree problem : Translation of both the 1926 papers, comments, history. *Discrete Mathematics*, 233:3–36, 2001. (Cité page 124.)

[158] Hieu T. Nguyen et Qiang Ji : Improved watershed segmentation using water diffusion and local shape priors. *In Proceedings of the 2006 IEEE Computer Society conference on Computer Vision and Pattern Recognition*, pages 985–992, Washington, DC, USA, 2006. IEEE Computer Society. (Cité page 116.)

[159] Mark Nitzberg, David Mumford et Takahiro Shiota : *Filtering, segmentation and depth*. Springer-Verlag, Berlin, 1993. (Cité pages 213 et 214.)

[160] Stanley Osher et James A. Sethian : Fronts propagating with curvature dependent speed : algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988. (Cité page 209.)

[161] Christos H. Papadimitriou et Mihalis Yannakakis : Optimization, approximation, and complexity classes. *In Proceedings of the 20th Annual Association for Computing Machinery Symposium on the Theory of Computing*, pages 229–234, New York, NY, USA, 1988. Association for Computing Machinery. (Cité page 24.)

[162] Kedar A. Patwardhan, Guillermo Sapiro et Marcelo Bertalmío : Video inpainting under constrained camera motion. *IEEE Transactions on Image Processing*, 16(2):545–553, 2007. (Cité page 231.)

[163] Theo Pavlidis : *Structural Pattern Recognition*, volume 1 de *Springer Series in Electrophysics*. Springer-Verlag, 1977. (Cité page 58.)

[164] Judea Pearl : Reverend bayes on inference engines : A distributed hierarchical approach. *In Proceedings of the American Association of Artificial Intelligence National Conference on AI*, pages 133–136, Pittsburgh, PA, 1982. (Cité page 210.)

[165] Judea Pearl : *Probabilistic reasoning in intelligent systems : networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988. (Cité page 210.)

[166] Frédéric Pighin, Jamie Hecker, Dani Lischinski, Richard Szeliski et David H. Salesin : Synthesizing realistic facial expressions from photographs. *In Proceedings of the 25th International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 75–84, New York, NY, USA, 1998. Association for Computing Machinery. (Cité page 236.)

[167] Marc Pollefeys : Self-calibration and metric 3d reconstruction from uncalibrated image sequences. *In Ph.D.* Katholieke Universiteit Leuven, 1999. (Cité page 234.)

[168] Marc Pollefeys : Visual 3d modeling from images. *In Tutorial notes*. 2002. (Cité pages 235 et 236.)

[169] Jean-Philippe Pons, Renaud Keriven et Olivier Faugeras : Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. *International Journal of Computer Vision*, 72(2):179–193, 2007. (Cité page 238.)

[170] Renfrey Burnard Potts : Some generalized order-disorder transformation. *In Proceedings of the Cambridge Philosophical Society*, volume 48, pages 106–109, 1952. (Cité page 181.)

[171] Robert Clay Prim : Shortest connection networks and some generalizations. *Bell System Tech J.*, pages 1389–1401, 1957. (Cité page 124.)

[172] Budirijanto Purnomo, Jonathan D. Cohen et Subodh Kumar : Seamless texture atlases. *In Proceedings of the 2004 Eurographics Symposium on Geometry Processing*, pages 65–74, New York, NY, USA, 2004. Association for Computing Machinery. (Cité page 236.)

[173] Claudio Rocchini, Paolo Cignoni, Claudio Montani et Roberto Scopigno : Acquiring, stitching and blending diffuse appearance attributes on 3D models. *The Visual Computer*, 18(3):186–204, 2002. (Cité page 236.)

[174] Jos B. T. M. Roerdink et Arnold Meijster : The watershed transform : definitions, algorithms and parallelization strategies. *Fundamenta Informaticae*, 41:187–228, 2000. (Cité pages 82 et 85.)

[175] Azriel Rosenfeld : *Picture processing by computer*. Academic Press, 1969. (Cité page 29.)

[176] Azriel Rosenfeld et Avinash C. Kak : *Digital picture processing*, volume 2. Academic Press, 1982. (Cité pages 58 et 67.)

[177] Rudolf Rothe : Zum Problem des Talwegs. *Sitzungsberichte der Berliner Math. Gesellschaft*, 14:51–69, 1915. (Cité page 75.)

[178] Carsten Rother, Sanjiv Kumar, Vladimir Kolmogorov et Andrew Blake : Digital tapestry. *In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 589–596, Washington, DC, USA, 2005. IEEE Computer Society. (Cité page 245.)

[179] 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. (Cité page 31.)

[180] Sébastien Roy et Ingemar J. Cox : A maximum-flow formulation of the n-camera stereo correspondence problem. *In Proceedings of the 6th International Conference on Computer Vision*, pages 492–502, Washington, DC, USA, 1998. IEEE Computer Society. (Cité page 206.)

[181] Punam K. Saha et Jayaram K. Udupa : Relative fuzzy connectedness among multiple objects : theory, algorithms, and applications in image segmentation. *Computer Vision and Image Understanding*, 82(1):42–56, 2001. (Cité page 125.)

[182] J. C. Saint-Venant : Considérations sur les surfaces à plus grande pente constante. *Bulletin de la société philomatique de Paris*, pages 24–30, 1852. (Cité page 75.)

[183] Huzur Saran et Vijay V. Vazirani : Finding k-cuts within twice the optimal. *Journal on Computing*, 24(1):101–108, 1995. (Cité page 158.)

[184] Daniel Scharstein et Richard Szeliski : A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3):7–42, 2002. (Cité page 206.)

[185] Alexander Schrijver : *Combinatorial optimization : polyhedra and efficiency*, volume A. Springer, 2003. (Cité pages 146 et 148.)

[186] Jean Serra, Corinne Vachier et Fernand Meyer : *Morphologie mathématique 1 : approches déterministes*, chapitre Nivellements, pages 173–200. Traité IC2, série signal et image. Hermès Science Publications, 2008. (Cité page 105.)

[187] James A. Sethian : *Level set methods : evolving interfaces in geometry, fluid mechanics, computer vision and materials science*. Cambridge University Press, New York, 1996. (Cité page 209.)

[188] James A. Sethian : A marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Science*, 93:1591–1595, 1996. (Cité page 209.)

[189] James A. Sethian : *Level set methods and fast marching methods : evolving interfaces in computational geometry, fluid mechanics, computer vision and materials science*. Cambridge University Press, 1999. (Cité page 209.)

[190] Alla Sheffer et John C. Hart : Seamster : inconspicuous low-distortion texture seam layout. *In Proceedings of the 2002 conference on Visualization*, pages 291–298, Washington, DC, USA, 2002. IEEE Computer Society. (Cité page 237.)

[191] Jianbo Shi et Jitendra Malik : Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 2000. (Cité page 210.)

[192] Daniel Dominic Sleator : An O(nm log(n)) algorithm for maximum network flow. Rapport technique STAN-CS-80- 831, Computer Science Department, Stanford University, 1980. (Cité page 153.)

[193] Daniel Dominic Sleator et Robert Endre Tarjan : A data structure for dynamic trees. *Journal of Computer and System Sciences*, 26(3):362–391, 1983. (Cité page 153.)

[194] Pierre Soille : Morphological image compositing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(5):673–683, 2006. (Cité page 245.)

[195] Jian Sun, Lu Yuan, Jiaya Jia et Heung-Yeung Shum : Image completion with structure propagation. *Proceedings of the 32nd International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 24(1):861–868, 2005. (Cité page 214.)

[196] Richard S. Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Vladimir Kolmogorov, Aseem Agarwala, Marshall F. Tappen et Carsten Rother : A comparative study of energy minimization methods for Markov random fields. *In Proceedings of the 9th European Conference on Computer Vision*, volume 3952, pages 16–29, Graz, Austria, 2006. Springer-Verlag. (Cité page 210.)

[197] Marshall F. Tappen et William T. Freeman : Comparison of graph cuts with belief propagation for stereo using identical mrf parameters. *In Proceedings of the 9th IEEE International Conference on Computer Vision*, volume 02, pages 900–907, Los Alamitos, CA, USA, 2003. IEEE Computer Society. (Cité page 210.)

[198] Robert Endre Tarjan : Efficiency of a good but not linear set union algorithm. *Journal of the Association for Computing Machinery*, 22(2):215–225, 1975. (Cité page 124.)

[199] Alan Mathison Turing : On computable numbers, with an application to the entscheidungsproblem. *In Proceedings of the London Mathematical Society*, volume 42. London Mathematical Society, 1937. (Cité page 22.)

[200] Alan Mathison Turing : On computable numbers, with an application to the entscheidungsproblem. a correction. *In Proceedings of the London Mathematical Society*, volume 43. London Mathematical Society, 1937. (Cité page 22.)

[201] Jayaram K. Udupa, Punam K. Saha et Roberto de Alencar Lotufo : Relative fuzzy connectedness and object definition : theory, algorithms, and applications in image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(11): 1485–1500, November 2002. (Cité page 125.)

[202] Jayaram K. Udupa et Supun Samarasekera : Fuzzy connectedness and object definition : theory, algorithms, and applications in image segmentation. *Graphical Models and Image Processing*, 58(3):246–261, 1996. (Cité page 125.)

[203] Corinne Vachier et Fernand Meyer : The viscous watershed transform. *Journal of Mathematical Imaging and Vision*, 22(2-3):251–267, 2005. (Cité page 116.)

[204] Luiz Velho et Jonas Jr Sossai : Projective texture atlas construction for 3D photography. *The Visual Computer*, 23(9):621–629, 2007. (Cité page 236.)

[205] Luc Vincent : Morphological grayscale reconstruction in image analysis : applications and efficient algorithms. *IEEE Transactions on Image Processing*, 2(2):176–201, 1993. (Cité page 111.)

[206] Luc Vincent et Pierre Soille : Watersheds in digital spaces : an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991. (Cité pages 75, 77, 82 et 105.)

[207] Martin J. Wainwright, Tommi S. Jaakkola et Alan S. Willsky : MAP estimation via agreement on (hyper)trees : message-passing and linear programming approaches. Rapport technique UCB/CSD-03-1269, EECS Department, University of California, Berkeley, August 2003. (Cité page 210.)

[208] Li-Yi Wei et Marc Levoy : Fast texture synthesis using tree-structured vector quantization. *In Proceedings of the 27th International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 479–488, New York, NY, USA, 2000. Association for Computing Machinery. (Cité page 214.)

[209] Hassler Whitney : Congruent graphs and the connectivity of graphs. *American Journal of Mathematics*, 54:150–168, 1932. (Cité page 29.)

[210] Gerhard Winkler : *Image analysis, random fields and dynamic Monte Carlo methods*. Springer-Verlag, 1995. (Cité page 181.)

[211] Mihalis Yannakakis, Paris C. Kanellakis, Stavros S. Cosmadakis et Christos H. Papadimitriou : Cutting and partitioning a graph after a fixed pattern (extended abstract). *In Proceedings of the 10th Colloquium on Automata, Languages and Programming*, pages 712–722, London, UK, 1983. Springer-Verlag. (Cité page 158.)

[212] Charles T. Zahn : Graph-theoretical methods for detecting and describing gestalt clusters. *Transactions on Computers*, C-20(1):68–86, 1971. (Cité page 124.)

[213] Eugene Zhang, Konstantin Mischaikow et Greg Turk : Feature-based surface parameterization and texture mapping. *Proceedings of the 32nd International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 24(1):1–27, 2005. (Cité page 237.)

[214] Yunjun Zhang, Jiangjian Xiao et Mubarak Shah : Motion layer based object removal in videos. *In Proceedings of the 7th IEEE Workshop on Application of Computer Vision*, volume 1, pages 516–521, Washington, DC, USA, 2005. IEEE Computer Society. (Cité page 231.)

[215] Song Chun Zhu, Yingnian Wu et David Mumford : Filters, random fields and maximum entropy (FRAME) : towards a unified theory for texture modeling. *International Journal of Computer Vision*, 27:1–20, 1998. (Cité page 214.)