REALISATION DE LA LIGNE DE PARTAGE DES EAUX PAR FILE D'ATTENTE HIERARCHIQUE PARALLELE

Etude algorithmique

Auteurs: **BEUCHER Serge**
LEMONNIER Fabrice
SASPORTAS Raphaël

17 Juin 1997

1. Introduction

La ligne de partage des eaux (en abrégé LPE) est une transformation largement utilisée depuis une quinzaine d'années en segmentation d'image. Elle présente, par rapport aux techniques de segmentation concurrentes, de nombreux avantages qui expliquent son succès. Parmi ces avantages, on peut citer, sans que cette liste soit exhaustive, les caractéristiques suivantes :
- c'est une transformation facile à comprendre, travaillant directement sur l'image sans passer dans le domaine spectral ou par des représentations complexes.
- c'est une transformation qui vient avec son mode d'emploi. La LPE permet en effet de séparer en deux étapes distinctes, la tâche de désignation des objets à segmenter de la tâche de segmentation et de délimitation proprement dite.
- la LPE est une transformation non paramétrique. Il n'y a nul besoin de fixer les valeurs de nombreux paramètres pour la réaliser.
- cette transformation s'emploie aussi bien sur les images fixes que sur les séquences, sur les images à niveaux de gris que sur les images couleur ou multi-spectrales, sur les images 2D ou 3D. C'est en particulier dans ce dernier champ d'application qu'elle a su montrer son efficacité.
- la LPE se prête bien aux techniques de segmentation hiérarchique. Les régions obtenues par un premier niveau de segmentation peuvent être traitées à nouveau par LPE afin de rassembler les régions homologues.

Cependant, à côté de ces indéniables qualités qui expliquent que ses domaines d'utilisation ne cessent de s'étendre, elle présente quelques inconvénients au premier rang desquels se situe sa relative lenteur couplée à certains pièges liés à sa mise en oeuvre algorithmique. Ces difficultés ont amené, depuis déjà quelques années, le CMM a s'intéresser à des améliorations d'algorithmes qui ont permis des gains de vitesse d'un facteur 100 par rapport aux implantations initiales. Ces gains ne sont malheureusement pas encore suffisant si l'on envisage l'emploi de la LPE dans les traitements temps réel. C'est pourquoi, le CMM en collaboration avec THOMSON - CSF OPTRONIQUE s'efforce dans le cadre d'un contrat financé par la DGA -

DRET(contrat N°95-520) de définir des algorithmiques et des architectures parallèles qui permettrait d'atteindre des vitesses de traitement beaucoup plus grandes.
Ce rapport intermédiaire décrit les travaux de nature algorithmique qui ont été entrepris dans le cadre de ce contrat. Ces travaux s'efforcent de définir de nouvelles approches parallèles à partir d'algorithmes existant déjà, en particulier les files d'attente hiérarchiques. Diverses pistes ont été explorées. Avant de les décrire en détail, on rappellera la définition de la LPE ainsi que ses deux grands modes d'utilisation, selon qu'elle est contrôlée par marqueurs ou non. Les algorithmes classiques de LPE seront également rappelés, car ils permettent de mettre le doigt sur différents problèmes de fond relatifs à cette transformation (notamment son caractère non local), problèmes qui influencent la vitesse de réalisation et l'exactitude du résultat final. On décrira ensuite la file d'attente hiérarchique (FAH), l'algorithme qui sert de point de départ aux développements actuels. Ces développements visent à paralléliser l'algorithme par FAH. Plusieurs solutions seront décrites : découpage de l'image en imagettes et traitement parallèle de ces imagettes, traitement en parallèle de plusieurs pixels sur la file d'attente, traitement en parallèle de marqueurs. Ces différentes solutions ne sont pas incompatibles. Elles peuvent être combinées, d'autant qu'on s'efforcera de montrer au cours de ce rapport que les performances dépendent fortement de la nature des images traitées et qu'une solution unique ne saurait être la plus efficace. On essaiera de comparer les différentes solutions proposées et de donner une idée des gains de vitesse envisageables, bien que ces estimations n'aient pas encore été testées sur des cas concrets. Ces tests seront réalisés à la suite de l'intégration de ces différentes approches dans l'environnement de développement Ptolemy. Cet environnement, développé par l'Université de Berkeley, permet de valider la fonctionnalité d'algorithmes parallèles.

2. Définition, construction et utilisation de la LPE

2.1 Définition

Soit f une fonction numérique quelconque, qui peut être, par exemple, la représentation d'une image à niveaux de gris. Pour introduire la ligne de partage des eaux de f, notée LPE(f), nous allons considérer simplement la surface topographique limitée par le sous-graphe G(f) de f. Cette frontière de G(f) présente un certain nombre de structures topographiques caractéristiques : dômes, vallées, lignes de crêtes ou de thalwegs, etc.. Parmi ces structures, deux nous intéressent plus particulièrement : les minima régionaux et les bassins versants de la topographie. Les minima de f sont les composantes connexes de la surface topographique formant des creux ou cuvettes Figure 1: Minima et plateaux d'une fonction.

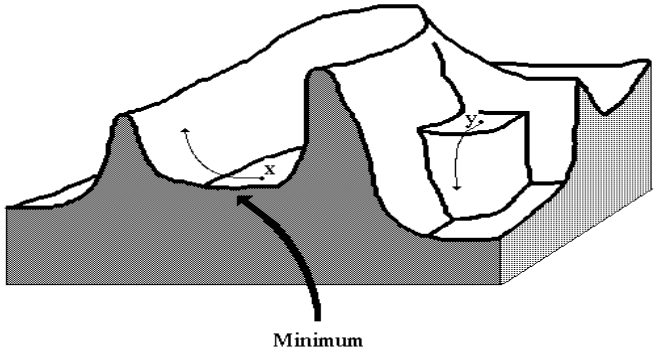


Figure 1: Minima et plateaux d'une fonction

Imaginons donc que cette surface topographique soit trouée aux emplacements des minima. Plongeons alors lentement cette surface dans un lac (étendue d'eau supposée infinie pour la commodité de l'expérience). l'eau va passer par les trous en commençant par ceux qui percent les minima les plus profonds et va progressivement inonder le relief. A tout moment de l'inondation, les différents lacs délimités sur la topographie seront à la même altitude (Figure 2). Ce fait est fondamental, nous y reviendrons par la suite.

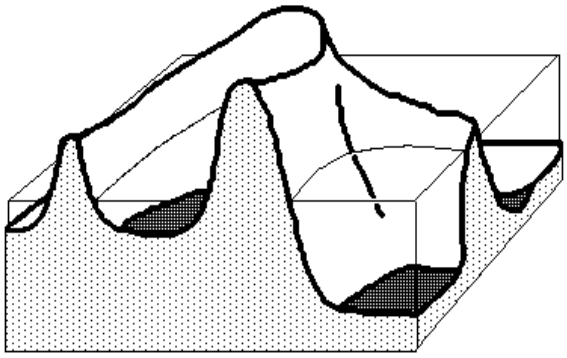


Figure 2: Inondation de la surface topographique définie par une fonction

Supposons de plus que l'on empêche les eaux provenant de lacs différents (donc de minima différents) de se mélanger en construisant sur la surface topographique un barrage toutes les fois où une telle éventualité pourrait se produire (Figure 3).

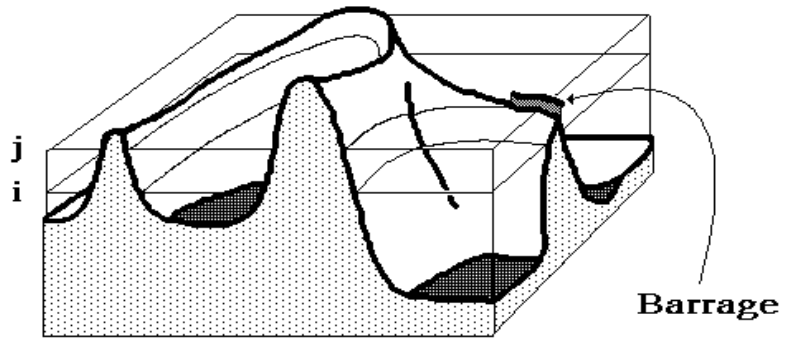


Figure 3: Construction d'un barrage (ligne de partage des eaux) entre les différents bassins versants

Lorsque la totalité de la surface topographique aura été engloutie, seuls les barrages émergeront, délimitant des lacs en nombre égal au nombre de minima de la fonction f. Ces barrages constituent ce qu'on appelle la ligne de partage des eaux de f. Quant aux lacs, ce sont les bassins versants associés aux minima de f (Figure 4).

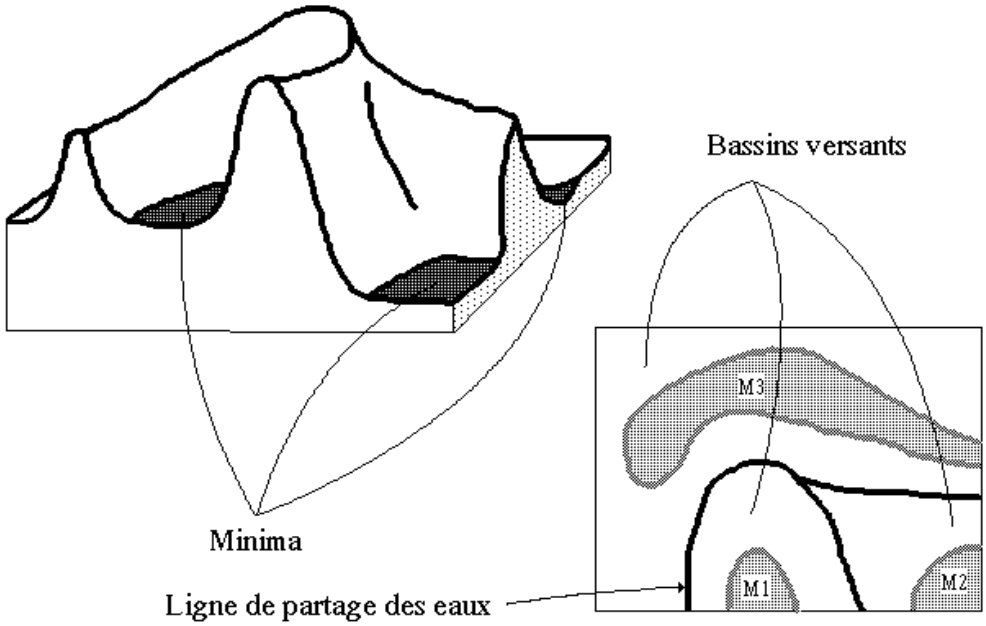


Figure 4: Ligne de partage des eaux (LPE) et bassins versants d'une fonction

On remarque immédiatement qu'il n'est pas nécessaire de faire apparaître réellement

les lignes de barrage. On pourrait imaginer de colorier différemment les eaux appartenant à des bassins versants différents. Cette opération s'assimile à un étiquetage et les lignes de partage des eaux sont alors d'épaisseur nulle. Ces deux variantes conduisent à des algorithmes différents. L'algorithme de construction de la LPE classique appartient à la première variante. On verra que l'algorithme par file d'attente utilisé comme point de départ de l'étude algorithmique procède de la seconde variante, en produisant des lignes de partage des eaux d'épaisseur nulle.

2.2 Construction de la LPE : l'algorithme classique

Cette définition de la ligne de partage des eaux en termes d'inondation présente également l'avantage d'être opératoire et de fournir un algorithme direct pour sa construction. Cet algorithme est basé sur la reconstruction des seuils successifs de la fonction f à l'aide d'une transformation morphologique appelée squelette par zones d'influence géodésique (SKIZ géodésique). Décrivons-le à l'aide d'un exemple.
Soit f une fonction digitalisée, et désignons par Z_i (f) l'ensemble des points x d'altitude inférieure ou égale à i.

$$Z_i(f)=\{x : f(x) \leq i\}$$

Considérons la plus petite altitude i_0 correspondant à un seuil $Z_{i0}(f)$ non vide. $Z_{i0}(f)$ peut avoir plusieurs composantes connexes, chacune d'elles étant alors par définition un minimum régional de f. Examinons alors le seuil $Z_{i0+1}(f)$ immédiatement supérieur. Ce dernier seuil contient évidemment le précédent. Soit Z, une composante connexe de $Z_{i0+1}(f)$. Il y a trois relations possibles entre Z et $Z_{i0}(f)$. (**Erreur ! Source du renvoi introuvable.**).

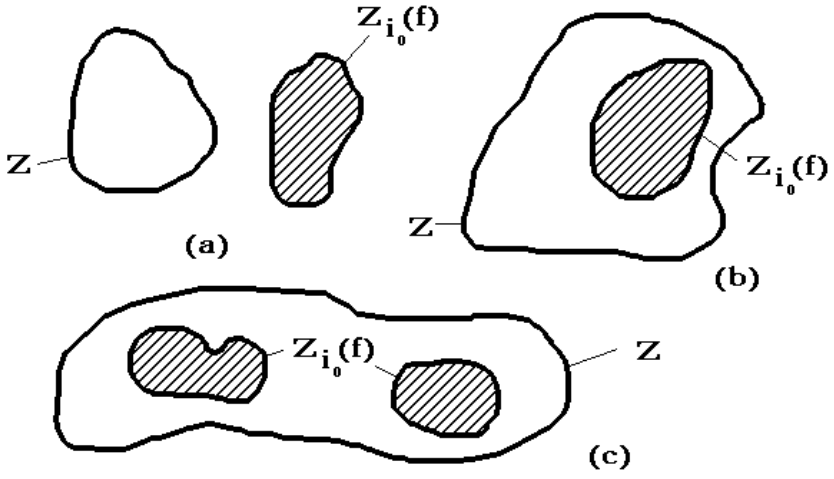


Figure 5: Relations entre les composantes connexes de deux seuils successifs d'une fonction

- Ou bien, $Z \cap Z_{i0}(f)$. Dans ce cas, Z est un minimum régional de f à l'altitude i_0.
- Ou encore, $Z \cap Z_{i0}(f)$ est non vide et connexe. Dans ce cas, Z représente le niveau

(i_0+1) du lac produit par l'inondation du minimum régional $Z \cap Z_{i0}(f)$.

- Enfin $Z \cap Z_{i0}(f)$ peut être non vide et formé de plusieurs composantes connexes. Dans ce cas, Z est la réunion des eaux provenant des différents minima régionaux composant $Z \cap Z_{i0}(f)$. Comme cette jonction n'est pas autorisée, il faut donc construire la ligne de partage des eaux séparant ces différents lacs.
Pour cela, on construit les zones d'influence géodésiques de $Z \cap Z_{i0}(f)$ dans Z (Figure 6).

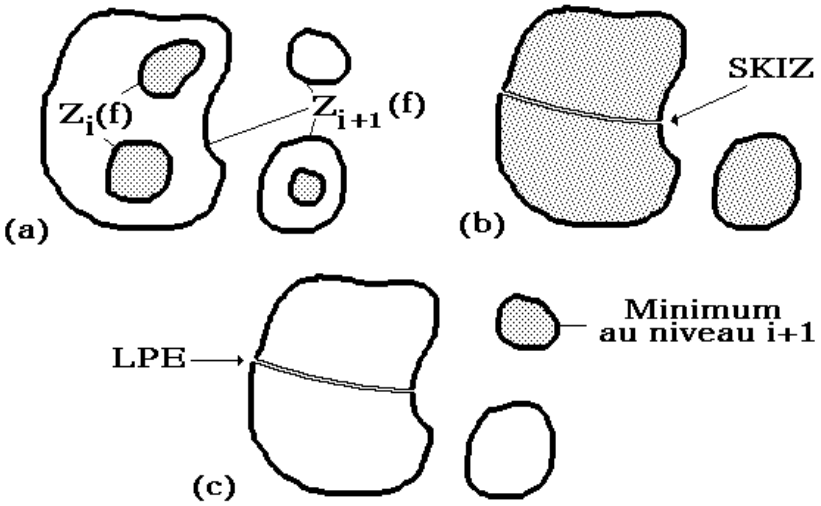
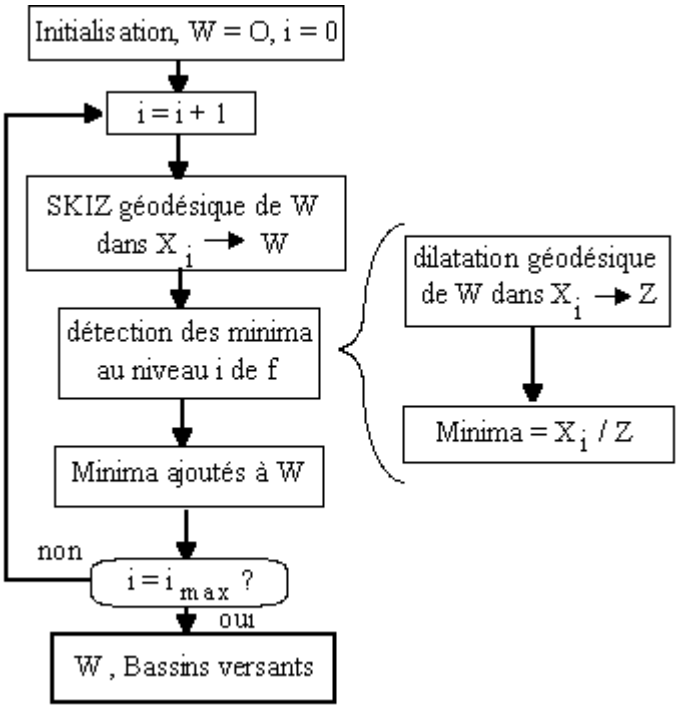


Figure 6: Construction de la LPE par SKIZ géodésique - étape initiale (a), SKIZ géodésique du seuil i dans le seuil i+1 (b), ajout des minima à ce niveau (c)

Une zone d'influence d'une composante connexe de $Z \cap Z_{i0}(f)$ est constituée des points de Z plus proches au sens de la distance géodésique de cette composante connexe que de tout autre composante connexe de $Z \cap Z_{i0}(f)$. Chaque zone d'influence constitue alors un bassin versant, ou du moins sa restriction au niveau $i_0 + 1$, associé à chaque minimum régional (composante connexe) de $Z \cap Z_{i0}(f)$.

Reprenons alors la totalité du seuil $Z_{i0+1}(f)$. Comme ce qui vaut pour une composante connexe de $Z_{i0+1}(f)$. vaut pour toutes, les bassins versants de f au niveau i_0+1 seront constitués des zones d'influence géodésiques de $Z_{i0}(f)$ dans $Z_{i0+1}(f)$ auxquelles viennent s'ajouter les minima régionaux au niveau i_0 +1, c'est-à-dire les composantes connexes de $Z_{i0+1}(f)$. d'intersection vide avec $Z_{i0}(f)$.
Il suffit alors de réitérer cette procédure de construction pour les niveaux i_0 +2, i_0+3, etc.. De façon plus formelle, on peut décrire cet algorithme à l'aide de l'ordinogramme suivant (f sera supposée prendre ses valeurs entre 0 et N).



A la fin de la procédure, W représente les bassins versants de f, et LPE(f)=W^C.

2.3 La LPE contrôlée par marqueurs

Dans la définition classique de la LPE, l'inondation de la surface topographique engendrée par la fonction f se fait à partir des minima de cette fonction. Cependant, rien n'interdit de construire une LPE où l'inondation ne serait plus amorcée par les minima mais par un ensemble quelconque de sources d'inondation. En reprenant l'analogie précédente, au lieu de percer la surface topographique à l'aplomb des minima de f, on peut le faire à l'aplomb de chaque composante connexe d'un ensemble quelconque M. Cet ensemble est appelé ensemble marqueur. L'inondation s'effectuant alors à partir de chaque composante connexe générera autant de bassins versants correspondants. Par rapport à l'algorithme précédent, on remarquera que l'eau, bien que restant à tout instant au même niveau peut être amenée à se déverser dans des cuvettes non marquées (Figure 7). En reprenant l'algorithme de construction seuil par seuil précédemment utilisé, cette LPE est paradoxalement plus simple à mettre en oeuvre.
L'initialisation de l'algorithme consiste à prendre l'ensemble M des marqueurs comme initiateur des bassins versants :

$$W_0=M$$

Puis l'inondation au niveau i des bassins versants W_{i+1} s'effectue par squelette par zones d'influence géodésique des bassins versants au niveau i dans l'espace $Z_{i+1}(f)\cup M$. La Figure 8 illustre l'algorithme.

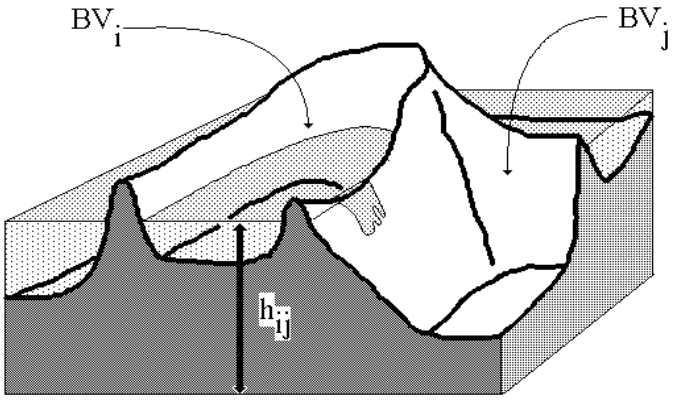


Figure 7: Débordement d'un bassin versant actif (marqué) dans un bassin versant non marqué

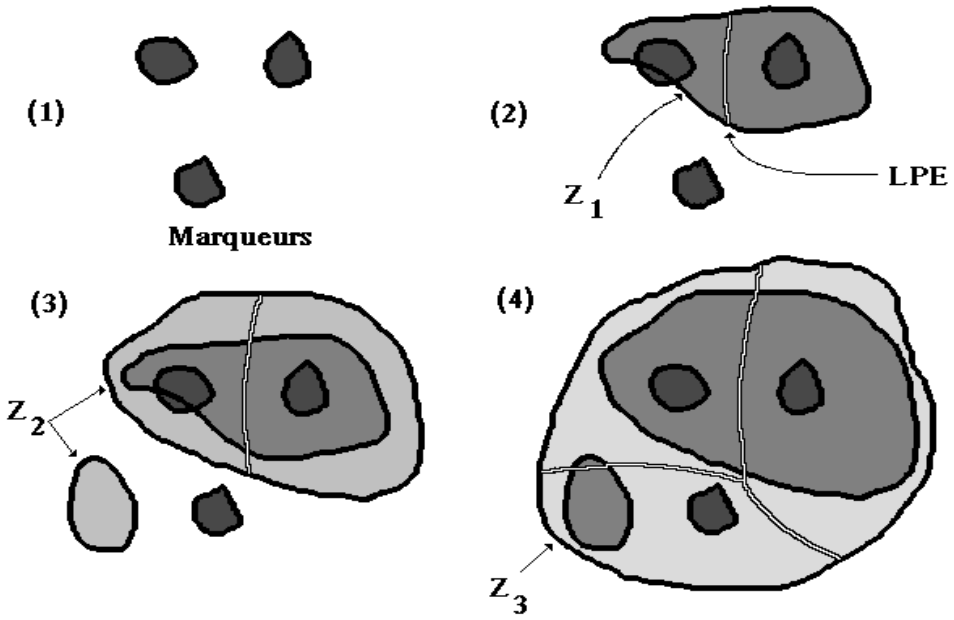


Figure 8: Algorithme de ligne de partage des eaux avec marqueurs imposés. Inondation des sections successives de la fonction

Deux différences essentielles existent entre cet algorithme et celui de la LPE classique: d'abord le SKIZ géodésique s'effectue dans $Z_{i+1}(f) \cup M$ et non dans $Z_{i+1}(f)$. En effet, W_i doit être inclus dans l'espace géodésique. Or comme :

$$M = W_0 \subset W_i$$

on n'est pas assuré que W_i soit inclus dans Z_i (f), d'où l'adjonction de M à ce niveau

de seuil. La deuxième différence est qu'on n'adjoint pas à W_i les minima de f apparus au niveau i. En effet, ces minima ne sont plus les sources de l'inondation. Cette différence explique également pourquoi cet algorithme est plus simple que l'algorithme classique. En effet ce dernier combine à la fois la construction de la LPE à chaque niveau de seuil et l'extraction des éventuels minima à ce niveau. L'algorithme classique apparaît alors comme un cas particulier de la LPE contrôlée par marqueurs. Il suffit en fait de détecter les minima de la fonction f et de prendre ces minima comme ensemble marqueur M. C'est pourquoi, les algorithmes de LPE analysés dans cette étude sont uniquement des algorithmes de LPE contrôlées par marqueurs.

2.4 Caractéristiques fondamentales de la LPE

Certaines caractéristiques fondamentales de la LPE tant au niveau de sa définition que de sa réalisation doivent être soulignées. Ce sont essentiellement le caractère non local de la transformation et la hiérarchie particulière engendrée par le processus d'inondation notamment sur les plateaux.

2.4.1 La non-localité de la LPE

Une ligne de partage des eaux est un objet non local. Cela signifie qu'il est impossible d'affirmer qu'un pixel donné d'une image appartient à la LPE par une simple observation locale de son environnement (même si on étend l'analyse au delà de ses voisins immédiats). La meilleure preuve de cette impossibilité est que l'on peut construire relativement facilement des configurations de voisinage d'un pixel de taille aussi grande que l'on veut telles que l'une fera du pixel central un point de la LPE et pas l'autre. La seule façon d'affirmer sans ambiguïté qu'un pixel appartient à la LPE est de constater que ce pixel sépare plusieurs eaux provenant de sources différentes. Il faut donc propager l'inondation. Ce caractère non local de la LPE explique également pourquoi cette ligne ne suit pas obligatoirement les lignes de crête de la surface topographique dessinée par la fonction f. C'est même parfois le contraire dans le cas par exemple de structures en boutonnière (Figure 9).

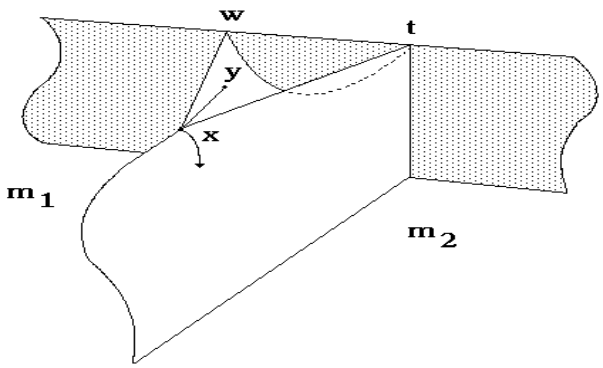


Figure 9: Structure en demi-boutonnière - La LPE passe par le thalweg entre les deux crêtes

On peut noter que le caractère non local de la LPE n'est pas complètement géré par les algorithmes présentés plus haut. En effet, le SKIZ géodésique est réalisé par le biais de transformations morphologiques locales (des épaississements homotopiques) et l'itération de ces transformations ne conduit pas nécessairement à une véritable LPE séparant des bassins versants différents. C'est le cas par exemple de la fonction présentée à la Figure 10. L'arc AB sépare le même bassin versant. Une telle ligne de partage est appelée ligne de partage locale.

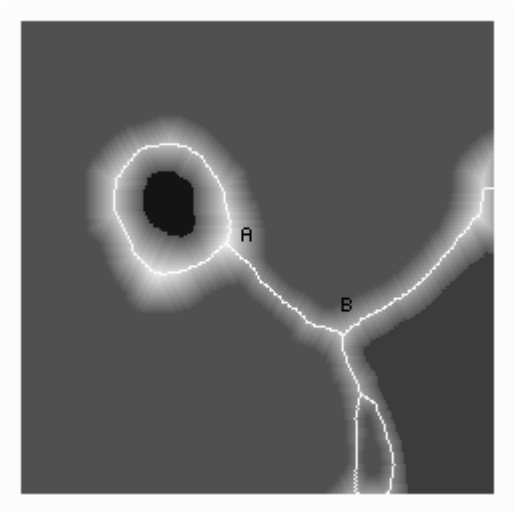


Figure 10: Ligne de partage locale engendrée pendant la construction de la LPE

Cette caractéristique fondamentale de la LPE constitue un problème majeur pour sa parallélisation. En effet, la parallélisation d'algorithmes consiste souvent à réaliser en même temps un certain nombre de transformations locales, l'union de ces transformations fournissant le résultat recherché sur l'ensemble de l'image.
Il existe cependant une classe particulière de fonctions où l'observation locale des configurations de pixels permet de mettre en évidence des points appartenant à la LPE (points-selles). Ces fonctions ont la propriété de ne pas posséder de zones plates. D'autre part, il est possible par le biais d'une opération appelée fléchage de transformer n'importe qu'elle fonction en une fonction sans zones plates ayant même LPE que la fonction initiale (Figure 11).
On pourrait penser alors qu'on tient là la solution au problème de la non-localité de la LPE. Malheureusement ce n'est pas le cas, car le fléchage est une opération qui doit être effectuée par le biais d'un processus de propagation.

2.4.2 Les hiérarchies dans le phénomène d'inondation

2.4.2.1 Un premier niveau de hiérarchie

Il existe dans la construction de la LPE plusieurs niveaux hiérarchiques qu'il importe de respecter quelque soit l'algorithme choisi, faute de quoi le résultat sera faussé. On a déjà vu que les eaux inondant les bassins versants sont toujours à la même altitude. On pourrait envisager de transgresser cette règle en permettant à certains bassins versants de se remplir plus vite que d'autres. Cependant, on a vu plus haut que la seule façon d'obtenir la LPE est de mettre en évidence le contact des eaux provenant

de sources d'inondation différentes. Dans l'hypothèse d'une vitesse d'inondation différente, cela signifie qu'il faudrait être capable d'arrêter momentanément l'inondation d'un bassin versant lorsqu'on se rend compte qu'il risque de déborder dans un autre (voir Figure 7) parce que l'eau passe par dessus la LPE qui les sépare. Or on a vu qu'on ne dispose d'aucun moyen de repérer cet éventuel débordement à cause de la non-localité de la LPE. Cette approche algorithmique est donc vouée à l'échec.

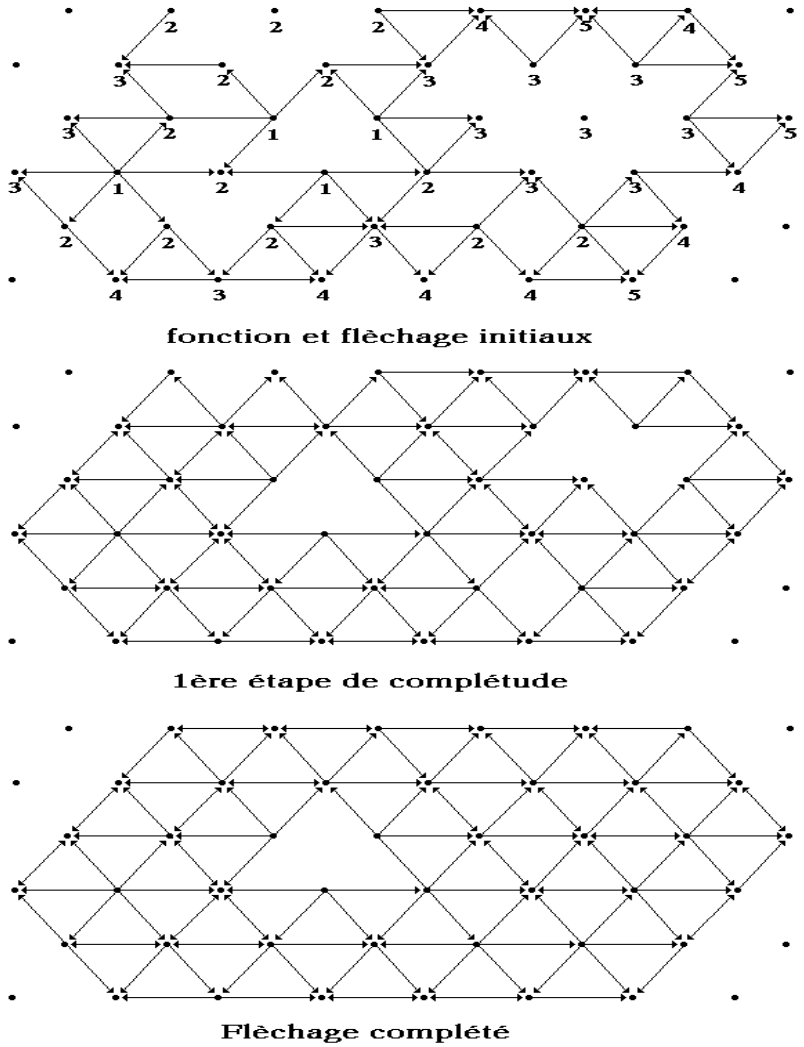


Figure 11: Fléchage d'une fonction - la complétude du fléchage élimine les zones plates

2.4.2.2 Propagation le long des plateaux

La deuxième caractéristique importante de la LPE concerne la propagation de l'inondation sur les zones plates de la surface topographique. L'algorithme classique est construit de manière à simuler une propagation à vitesse constante : l'eau arrive sur le bord descendant du plateau et inonde ce plateau en se propageant à vitesse constante à partir du bord. Le front d'inondation est à chaque instant et partout à la

même distance du bord et ceci pour l'ensemble des plateaux. Cette modélisation de l'inondation sur les zones plates est conforme à la réalité. Bien entendu, si un plateau est inondé par plusieurs sources, les règles de construction de la LPE doivent être respectées et celle-ci se construit sur le plateau en fonction des contacts des différentes eaux provenant des différents lacs adjacents au plateau. Les distances successives des fronts de propagation correspondent à la distance géodésique des bords descendants du plateau aux autres points dudit plateau (Figure 12).

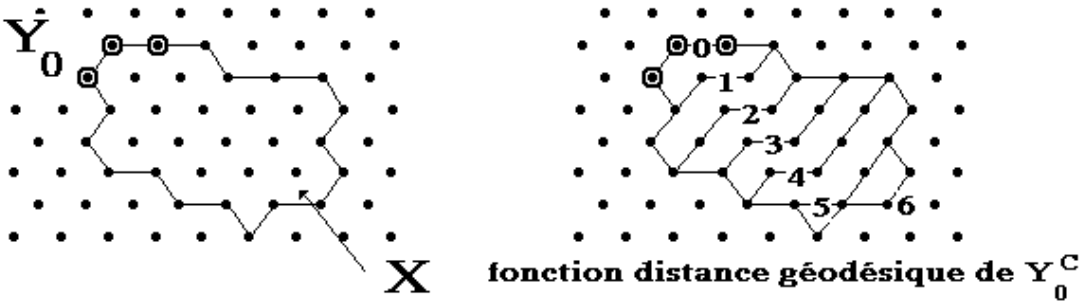


Figure 12: Plateau X, de bord descendant Y_0 - Distance géodésique définie sur le plateau X à partir des bords descendants

Ce modèle de l'inondation des plateaux engendre une nouvelle hiérarchie qui implique que les points du bords doivent tous être traités partout avant que les points à la distance 2 soient tous traités à leur tour, et ainsi de suite jusqu'à ce que tous les points du plateau aient été inondés. Comme les plateaux n'ont aucune raison d'avoir la même taille, on voit immédiatement que leur inondation ne s'achèvera pas en même temps. Cela signifie alors que les premiers plateaux inondés devront attendre que tous les plateaux d'une même hauteur aient été inondés pour que le processus puisse se poursuivre vers les points de la surface topographique de hauteur immédiatement supérieure. On pourrait envisager de changer la règle de propagation en remplaçant le modèle à vitesse constante par un modèle à débit constant : les points des plateaux seraient traités l'un après l'autre sans tenir compte de leur distance au bord descendant. La propagation le long des petits plateaux serait alors beaucoup plus rapide que sur les grands. Cette modélisation n'est pas intéressante pour plusieurs raisons :
- la règle de propagation sur les plateaux n'est pas la même que sur le reste de la surface topographique. Il semble difficile de justifier pourquoi une inondation à débit constant devrait être la règle sur les plateaux alors que ce n'est manifestement pas le cas sur le reste de la surface topographique puisque la hauteur de l'inondation est la même pour tous les bassins versants et ceci quelque soit leur volume.
- Cette règle induirait un positionnement différent de la LPE sur les plateaux qui a toutes les chances d'être biaisé.
- Le gain en vitesse serait illusoire puisque, quelque soit la règle d'inondation, il est indispensable d'attendre que l'ensemble des plateaux soient inondés pour poursuivre le processus. Or dans tous les cas de Figures, le nombre de points des plateaux à traiter reste inchangé.

2.4.3 Les autres biais de l'algorithmique

En dehors des erreurs grossières qui peuvent entacher la LPE si l'algorithme utilisé fait table rase des hiérarchies de l'inondation décrites plus haut, il existe aussi des biais engendrés par les transformations morphologiques utilisées et notamment par les transformations homotopiques servant à construire le SKIZ géodésique. Cette transformation utilise des opérateurs appelés épaississements réalisés à l'aide d'éléments structurants biphasés et anisotropes. Pour des raisons de simplicité algorithmique, ces éléments structurants ne sont pas utilisés en même temps mais direction par direction. Cela génère des biais comme l'illustre la Figure 13Figure 13où ce type de transformation est utilisée pour effectuer le SKIZ de deux points.

Figure 13: Biais dans la réalisation du SKIZ dû à l'usage d'épaississements non isotropes

Comme l'opération est itérative, les erreurs locales se propagent et peuvent produire des dérives considérables. La Figure 14 montre l'importance de ces variations dans la construction de la LPE selon que l'on utilise un algorithme isotrope (à gauche) ou direction par direction (à droite).

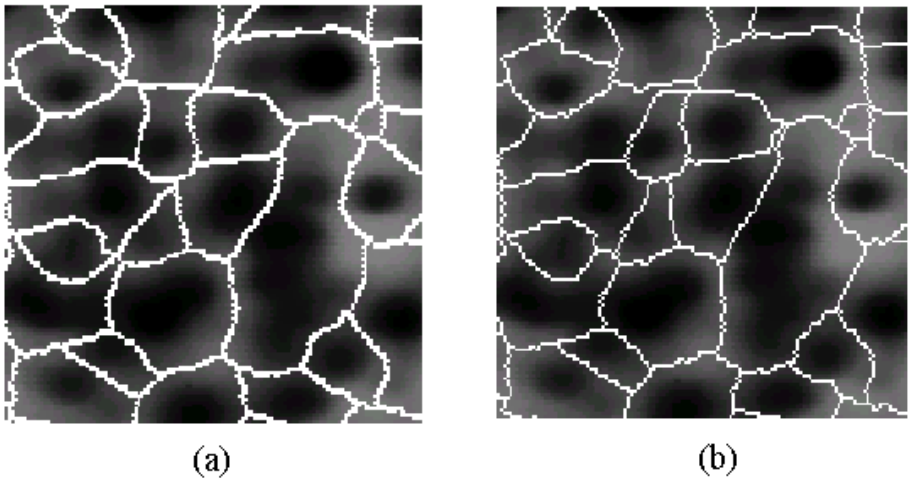


Figure 14: LPE exacte réalisée à l'aide d'opérateurs isotropes (a) et LPE classique biaisée (b)

Paradoxalement et contrairement aux erreurs qui peuvent se produire si on ne respecte pas les relations d'ordre imposées par l'inondation niveau par niveau puis sur les plateaux, ces défauts apparaissent car on introduit artificiellement ici un ordre de traitement des pixels alors qu'il devraient tous être traités en même temps. On reviendra sur ce problème lors de la présentation des algorithmes de parallélisation afin d'envisager des solutions pour le réduire voire même le supprimer. Remarquons enfin que la LPE d'une image digitale devrait en toute rigueur avoir une épaisseur variable selon la parité des configurations : selon que l'on veut matérialiser ou non cette LPE, son épaisseur pourrait être de 0, 1 ou 2 pixels. L'utilisation d'épaississements direction par direction permet d'avoir une LPE d'épaisseur constante. On peut se demander si cette simplification algorithmique mérite les défauts importants qu'elle peut induire.

2.5 Comment utiliser la LPE?

Après avoir décrit la LPE, après avoir passé en revue ses caractéristiques et ses pièges et avant d'introduire l'algorithme de FAH qui servira de base à l'approche parallèle, on peut illustrer brièvement l'utilisation de la LPE. On a vu en effet que les performances de cette transformation seront fortement dépendantes de la nature des images traitées. De plus, cet outil de segmentation vient avec son mode d'emploi. Segmenter une image consiste en effet à mettre en évidence un ensemble de marqueurs M désignant les objets à extraire dans l'image et une fonction f quantifiant la nature des transitions entre les différents objets. Muni de ces deux éléments, la segmentation consiste alors simplement à effectuer la LPE de f contrôlée par les marqueurs M (Figure 15).

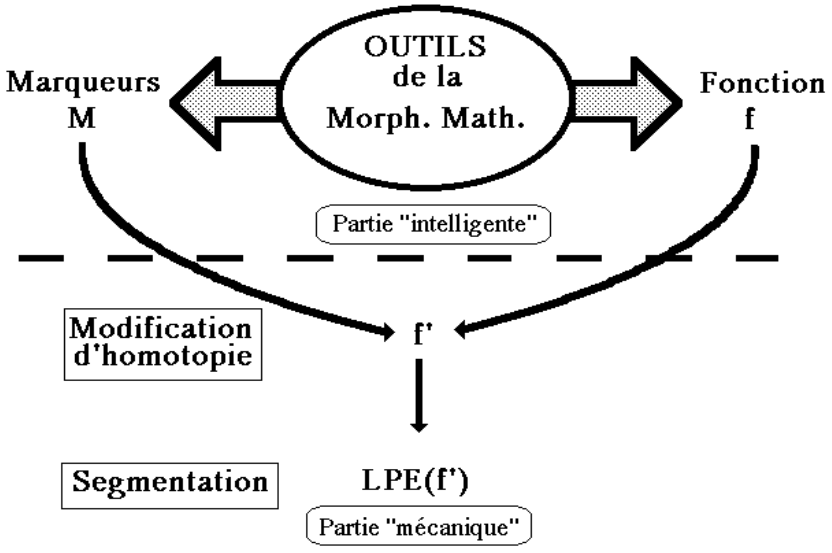


Figure 15: Principe général de la segmentation par LPE.

Très souvent (mais pas exclusivement) la fonction f utilisée est le module du gradient car les transitions entre les objets se caractérisent par des variations plus ou moins fortes de contraste. Les marqueurs M utilisés dépendent essentiellement de la nature et des propriétés de luminance, de couleur, géométrique ou topologiques des objets à segmenter.
On illustrera l'utilisation de la LPE sur un problème de segmentation automatique de la chaussée à partir d'une caméra embarquée dans un véhicule. La LPE du gradient de l'image originale (Figure 16) produit une forte sur-segmentation.

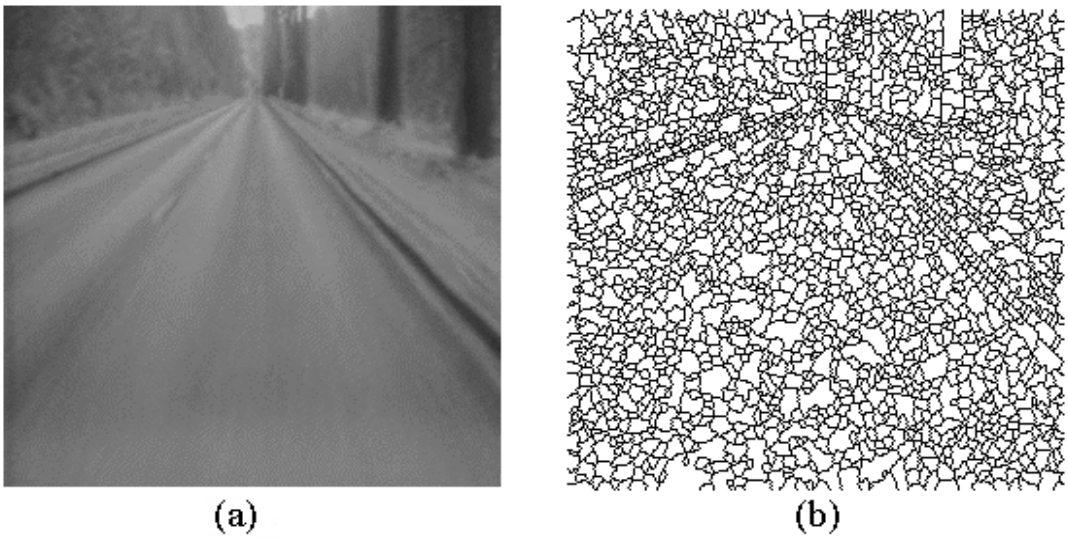


Figure 16: Image de la chaussée (a) et LPE de son gradient (b)

Mais si cette LPE est contrôlée par un ensemble de marqueurs (Figure 17a) désignant d'une part la chaussée et d'autre part l'extérieur, la segmentation obtenue est tout à fait satisfaisante (Figure 17b).

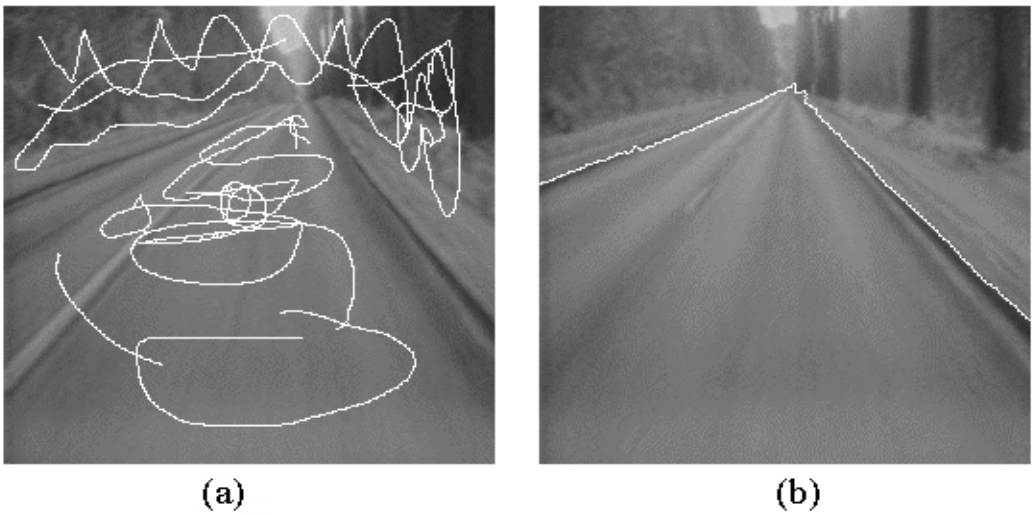


Figure 17: Marqueurs de la chaussée et de l'extérieur définis manuellement (a), LPE du gradient contrôlée par ces marqueurs (b)

Les marqueurs précédents ayant été introduits manuellement, il convient de définir une procédure automatique permettant de les générer. Cette procédure est basée sur une segmentation hiérarchique de la LPE initiale associée à une extraction du bassin versant situé en bas de l'image qui après simplification et filtrage fournira les marqueurs. La LPE finale sera identique à celle obtenue avec les marqueurs définis à la main (Figures 18a à 18d).

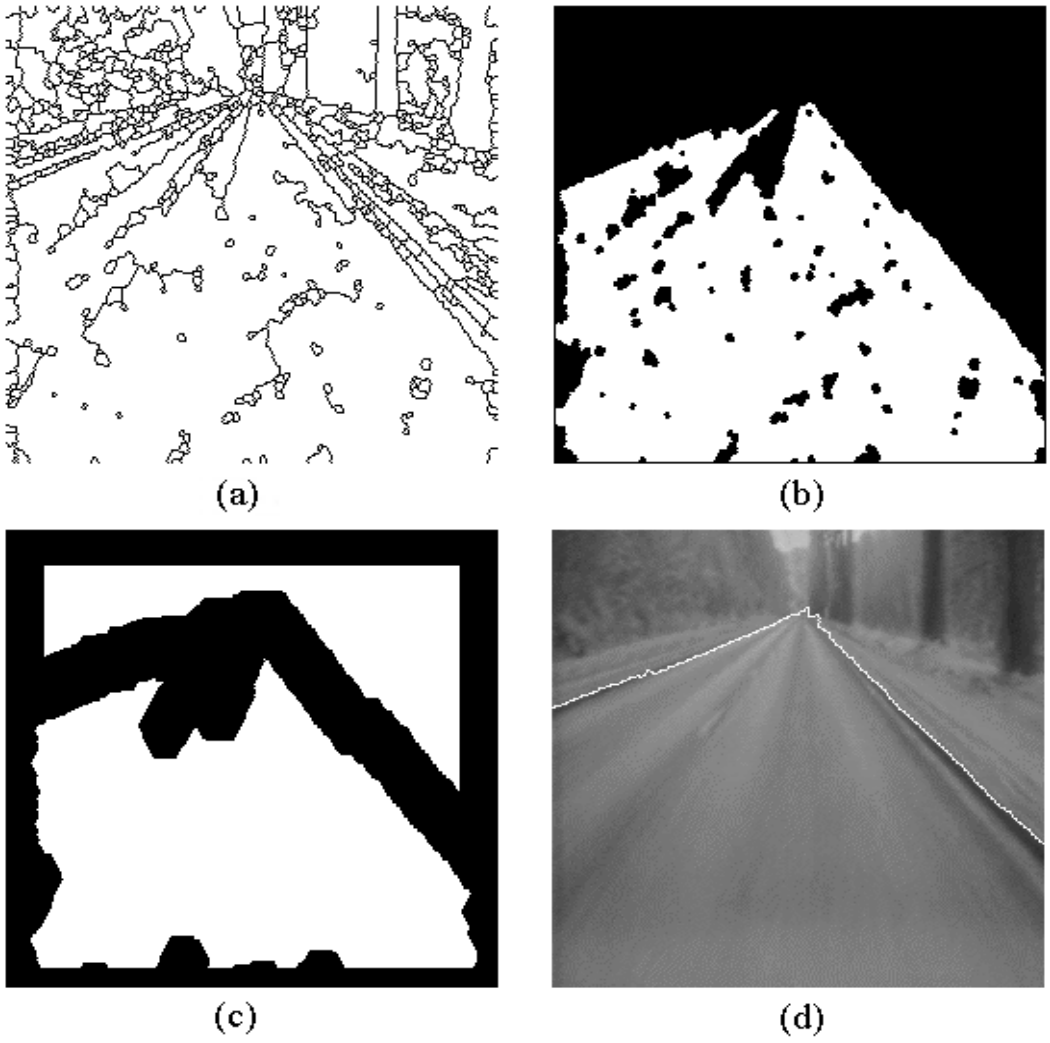


Figure 18: Première segmentation hiérarchique de l'image (a), extraction d'un marqueur de la chaussée (b), marqueur filtré et génération d'un marqueur extérieur (c), LPE du gradient contrôlée par les marqueurs précédents (d)

Cet exemple illustre plusieurs aspects de l'utilisation de la LPE. Très souvent, on a besoin à la fois de LPE de bas niveau et de LPE contrôlée par marqueurs. Si le gradient est une fonction qui présente à la fois beaucoup de minima et peu de plateaux, la LPE contrôlée par marqueur présente les caractéristiques inverses (l'inondation des cuvettes non actives est équivalente à la présence de plateaux). On verra que ces deux aspects antagonistes conditionnent énormément le rendement des solutions algorithmiques proposées pour la parallélisation.

3. Parallélisation de la LPE

3.1 Quelques solutions au problème de la vitesse de la LPE

L'algorithme classique est extrêmement lent. Il fonctionne en effet avec des opérateurs morphologiques qui travaillent sur toute l'image. Il a été écrit pour être réalisé sur une architecture de processeur basée sur un voisinage de traitement local et un balayage systématique de l'image. Tous les points de l'image sont traités à chaque itération. Or seuls les points venant d'être inondés et ceux susceptibles de l'être méritent de l'intérêt. Pour accélérer l'algorithme classique, diverses solutions ont été proposées. On peut réduire le nombre de niveaux de gris par des anamorphoses. Cela réduit le nombre d'itérations mais pas le nombre de points à traiter. On peut également sous-échantillonner l'image. Cette technique a été utilisée avec succès dans des cas bien précis. Elle ne saurait prétendre à être générale. Enfin d'autres approches algorithmiques ont été étudiées : fléchage, algorithmes récursifs et files d'attente hiérarchiques (FAH). La FAH constitue à l'heure actuelle la meilleure solution pour une implantation logicielle de la LPE. Ses performances sur des machines standards en font une alternative sérieuse à des processeurs dédiés. C'est pourquoi cette structure algorithmique a été choisie comme point de départ à la parallélisation de la LPE.

3.2 La file d'attente hiérarchique, présentation et fonctionnement

Une file d'attente hiérarchique peut être considérée comme une file d'attente multiple. Les jetons arrivent dans la file et sont traités par ordre de priorité. Chaque jeton est placé à la fin d'une file correspondant à son niveau de priorité : il sera traité après les jetons de même priorité qui sont arrivés avant lui. Dans la FAH, un seul jeton est traité à la fois. Dès qu'une file d'une priorité donnée est vide, elle est supprimée et elle ne sera jamais reconstituée dans la suite du processus. Si un jeton de priorité supérieure ou égale à la priorité de la file qui a été supprimée arrive, ce jeton sera placé à la fin de la file de priorité la plus élevée existant encore au moment de son arrivée. La spécification fonctionnelle d'une FAH se fait par le biais d'un nombre réduit d'items :
- la création de la FAH.
- la destruction d'une FAH.
- l'insertion d'un jeton au sommet d'une file de priorité donnée.
- le dépilage (ou traitement) consistant à fournir l'adresse et la priorité du jeton de plus haute priorité arrivé le premier (celui situé en bas de la pile de plus haute priorité).

3.2.1 Fonctionnement de la FAH

La Figure 19a montre comment une simple file d'attente fonctionne. Les jetons arrivent par le haut et sont retirés par le bas (structure FIFO "First In First Out"). La

Figure 19b montre comment une FAH fonctionne.

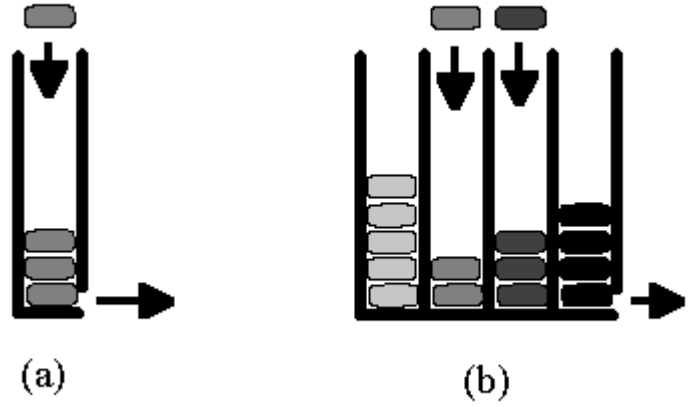


Figure 19: File d'attente simple (a) et file d'attente hiérarchique (b)

Une FAH est simplement une série de files d'attente simples. Chaque file d'attente simple a un niveau de priorité. Dans notre exemple la FAH a quatre niveaux de priorité et la file de plus forte priorité est à droite. Toutes les files sont ouvertes à leur sommet, ce qui signifie qu'à tout moment un jeton peut être inséré dans la file de priorité correspondante. Au contraire, seule la file de plus forte priorité peut être vidée. Le jeton qui peut être extrait est le jeton de plus forte priorité arrivé le premier dans la file. Dès que la file de plus forte priorité est vide, elle est supprimée et c'est la file de priorité immédiatement inférieure qui peut être vidée à son tour. La Figure 20 illustre la dernière caractéristique du fonctionnement d'une FAH : un jeton de forte priorité arrive après que la file de priorité correspondante ait disparue. Le jeton est alors inséré au sommet de la file courante de plus forte priorité.

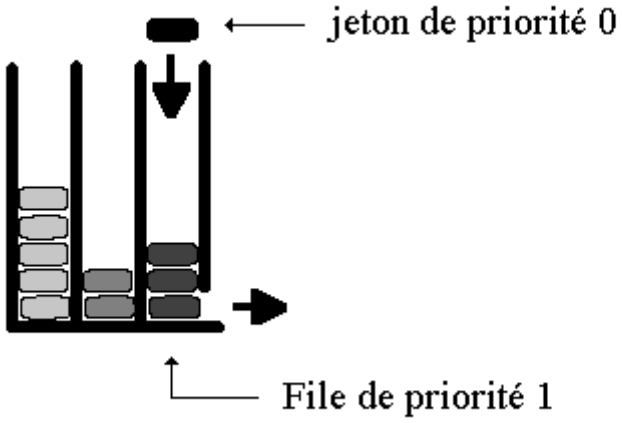


Figure 20: Prise en compte d'un jeton de priorité plus forte que la priorité de la file d'attente courante dans la FAH

3.2.2 Utilisation d'une FAH pour la LPE

La FAH peut être utilisée pour réaliser rapidement une LPE d'une fonction f contrôlée par un ensemble de marqueurs M. Chaque marqueur est étiqueté. Chaque région ou bassin versant en cours de construction conservera l'étiquette du marqueur qui l'aura généré. Un marqueur peut être scindé en plusieurs composantes connexes dès lors que chacune de ces composantes connexes a une étiquette commune. La réalisation de la LPE peut être effectuée selon deux variantes. Dans la première, les bassins versants se touchent. On désigne cette version par le terme de ZPE (zone de partage des eaux) car la frontière entre les bassins versants n'est pas définie sur les pixels mais passe entre eux dans la "zone" entre les pixels. Dans la deuxième variante, les frontières sont matérialisées par certains pixels de l'image. On ne s'est intéressé dans cette étude qu'à la première variante (ZPE).
L'algorithme de LPE par FAH comprend deux phases: une phase d'initialisation et une phase de fonctionnement.

3.2.2.1 L'initialisation

Une file d'attente hiérarchique est créée avec autant de niveaux de priorité qu'il y a de niveaux de gris dans l'image f. Associée à cette FAH se trouvent deux mémoires d'images: la première contient l'image f. C'est donc par essence la mémoire qui stocke les priorités (niveaux de gris) des pixels. La deuxième mémoire est la mémoire contenant les étiquettes des pixels traités. Elle est encore appelée mémoire-label et sera notée g. Ces deux mémoires sont initialisées au début du processus. La mémoire image ne sera pas modifiée en cours de traitement (elle est en lecture seule) tandis que la mémoire-label sera lue pour alimenter la FAH et écrite au cours du traitement pour mettre à jour les étiquettes. C'est elle qui contiendra les bassins versants étiquetés à la fin de la procédure.
La FAH est initialisée en stockant dans les files d'attente respectives les jetons correspondant aux pixels étiquetés dans l'image g ou plus simplement (ceci afin de réduire leur nombre) aux pixels appartenant aux frontières des régions étiquetés. Ces jetons transportent une seule information: les coordonnées du pixel correspondant dans f ou g.

3.2.2.2 Fonctionnement

Le fonctionnement de la FAH peut se résumer par le diagramme suivant:

Tant que la FAH n'est pas vide, faire:
{ - extraire un jeton x de la FAH
- déterminer les pixels voisins de x qui n'ont pas d'étiquette dans g.
- pour chaque voisin y non étiqueté, faire:
{ - assigner à y dans l'image g la même étiquette que x.
- insérer le jeton y dans la file d'attente de la FAH de priorité correspondant au niveau de gris de y dans f (si elle existe) ou à la file d'attente de plus forte priorité existant encore. }}

3.2.2.3 Illustration

La Figure 21 montre comment l'algorithme fonctionne sur un exemple

monodimensionnel très simple. On a représenté à gauche la surface topographique f ainsi que l'image label g. Dans cet exemple la source est simplement constituée par les minima de f. La partie droite montre l'évolution de la FAH.

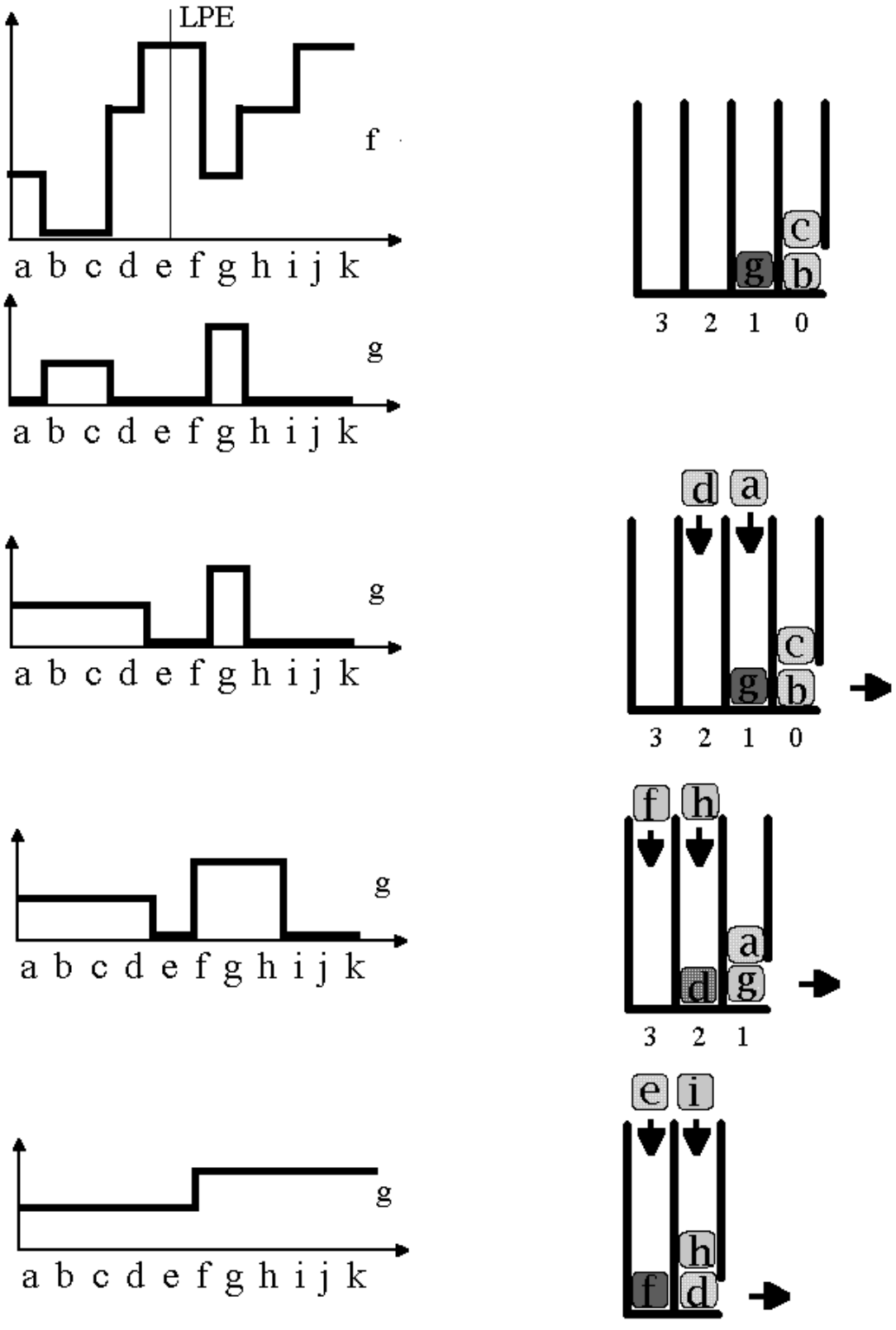


Figure 21: Exemple de fonctionnement de la FAH lors de la réalisation de la LPE

Initialisation

Une FAH est créée avec quatre niveaux de priorité correspondant aux quatre niveaux

de gris de l'image f. Les points-frontière des minima sont stockés dans la FAH avec une priorité correspondant à leur niveaux de gris. Les points b et c avec la priorité 0, le point g avec la priorité 1. L'image-label contient deux composantes connexes de valeurs respectives 1 et 2.

Construction de la LPE

Le point b de priorité maximale quitte la FAH. Son seul voisin sans étiquette est le point a. Ce point prend donc l'étiquette 1 (écrite dans la mémoire g) et est placé dans la file d'attente de priorité 1. Le jeton c est alors dépilé. Son seul voisin non étiqueté est d. d prend l'étiquette 2 et le jeton d de niveau de gris égal à 2 est placé dans le file de priorité 2. La file de priorité 0 étant alors vide est supprimée. Le prochain jeton quittant la file est le jeton g. Ses voisins f et h n'étant pas étiquetés, ils reçoivent l'étiquette 2 et sont stockés respectivement dans les files de priorité 3 et 2.

Le traitement de la FAH se poursuit. a est dépilé et n'empile aucun voisin. La file de priorité 1 étant alors vide, le dépilage de la file de priorité 2 commence par le traitement du jeton d qui empile son voisin e (étiqueté 1) suivi du dépilage du jeton h qui empile sur la même file d'attente son voisin i. i est alors dépilé à son tour. Ce jeton empile son voisin j sur la file de plus faible priorité. Cette file est alors traitée. Les jetons f puis e puis j sont dépilés, ce dernier étant le seul à empiler le jeton k. A la fin du traitement, l'image-label g contient les deux bassins versants associés aux deux marqueurs originaux.

3.2.2.4 Discussion

Cet algorithme est rapide puisque chaque pixel n'est traité qu'une fois contrairement à l'algorithme classique qui repasse sur la totalité de l'image à chaque itération. Cet algorithme respecte également les hiérarchies qui ont été mises en évidence précédemment. La montée uniforme de l'inondation est garantie par le fait qu'une file de priorité donnée n'est traitée qu'à partir du moment où les files de priorité supérieure sont vides. La hiérarchie de la propagation sur les plateaux est également respectée du fait que les premiers voisins des points des bords descendants des plateaux sont tous traités avant les deuxièmes voisins puisqu'ils sont empilés les premiers. Les deuxièmes voisins ne pourront être empilés que lorsque les premiers voisins seront dépilés. Cependant, cet algorithme comme l'algorithme classique basé sur des épaississements direction par direction produit un biais puisqu'il introduit un ordre de traitement arbitraire des pixels de même priorité. Ainsi dans l'exemple précédent, on aurait très bien pu empiler le jeton c avant le jeton b.

3.3 Parallélisation de la FAH

Comment accélérer la construction de la LPE en parallèlisant la FAH? Si on se réfère à nouveau au processus d'inondation qui constitue la définition opératoire de la LPE, on constate que ce processus est à la fois séquentiel et parallèle. Il est séquentiel car un ordre d'inondation doit être respecté à la fois dans la propagation le long du relief et des plateaux. Mais ce processus est également parallèle, car, à un niveau de priorité donné, ce sont tous les pixels appartenant à ce niveau qui se propagent en même temps. Or, c'est précisément ce parallélisme qu'il faut installer dans

l'algorithme de FAH si l'on veut accroître les vitesses de traitement. Il faut donc paralléliser le traitement des pixels de la file d'attente qui appartiennent au même niveau de priorité. Cette approche a un double avantage:
- elle seule permet de réaliser une LPE respectant les règles hiérarchiques précédemment énoncées.
- cette façon de faire supprime l'ordre de traitement artificiel des pixels introduits par les autres algorithmes (classique ou FAH). Elle est donc plus proche du phénomène naturel d'inondation et donc susceptible de réduire sinon d'éliminer les biais apparaissant dans les LPE actuelles.
Une fois ceci établi, il existe plusieurs façons de sélectionner les pixels ou les jetons de même niveau de priorité qui conduisent à plusieurs types d'architectures différentes et à diverses contraintes d'utilisation et de partage de la FAH, de la mémoire image et de la mémoire-label. On peut en effet:
- sélectionner les pixels selon leur position dans l'image. Cela revient à découper l'image en plusieurs imagettes et à traiter en même temps les pixels appartenant à des imagettes différentes.
- traiter en même temps les pixels appartenant à des sources d'inondation (marqueurs) différents.
- répartir sans considération sur leur position et leur sources d'inondation tous les pixels à traiter sur l'ensemble des processeurs disponibles.
Ce sont ces différentes approches que nous allons détailler dans la suite de ce document.
Signalons enfin qu'il est également possible d'accélérer le traitement en chercher à paralléliser (ou à "pipe-liner") le traitement réalisé par chaque processeur de la FAH sur chaque jeton.

4. Parallélisation de la LPE basée sur la FAH

4.1 Point de départ et Choix généraux

Comme nous l'avons dit précédemment, l'algorithme de zone de partage des eaux basé sur la FAH servira de point de départ à la parallélisation. Il s'agit de produire une image composée d'un ensemble de régions étiquetées séparées par une LPE d'épaisseur nulle.

Le nombre d'opérations à effectuer est nettement moins important pour la ZPE que pour la LPE. Dans le cadre d'une architecture multiprocesseur, l'implantation de la LPE nécessite une grande quantité d'échange entre les processeurs ce qui risque de ralentir considérablement le processus. En effet, il faut connaître les labels de tous les voisins du pixel dépilé pour pouvoir l'étiqueter. Inversement, il faut connaître le label du pixel dépilé pour savoir s'il faut empiler les pixels voisins non encore étiquetés.

Dans cette étude nous nous sommes donc concentrés sur la parallélisation de la ZPE.

Rappelons que nous avons postulé que la propagation des marqueurs se faisait à vitesse constante ce qui impose de propager les marqueurs à l'intérieur de chaque niveau de gris (plateaux) suivant la distance géodésique à partir des bords descendants.

4.2 Qu'est-ce qui est parallélisable ?

Nous avons déterminé que seuls deux types de parallélisation sont possibles :
- traiter les n pixels situés dans la file d'attente courante en même temps.
- paralléliser les étapes de traitement d'un pixel (dépiler, étiqueter, empiler).

L'avantage de ces deux techniques est qu'elles peuvent être combinées car la première approche permet de traiter plusieurs pixels en même temps alors que la deuxième approche permet d'accélérer le traitement de chaque pixel.

Nous allons détailler les différentes architectures envisagées pour réaliser ces deux types de parallélisation.

4.2.1 Parallélisation des n jetons situés dans la pile de niveau de priorité courant

4.2.1.1 Les trois types d'architectures envisagées

Nous avons envisagé trois voies de parallélisation (voir Figure 22) :
- un processeur par imagette : une zone de l'image de départ (imagette) est affectée à chaque processeur. Chaque processeur effectue le processus de propagation uniquement dans cette zone. Les trois mémoires (FAH, LABEL, Niveaux de Gris) sont distribuées sur l'ensemble des processeurs ;
- un processeur par marqueur : il y a une FAH par processeur. Dans chacune d'elles sont empilés les pixels ayant la même valeur de label (même marqueur). L'étape d'étiquetage se simplifie puisqu'il n'y a plus à déterminer la valeur du label du pixel courant (dépilé). Les processeurs n'ont pas de zone mémoire prédéterminée sur laquelle ils effectuent la propagation. L'image LABEL sera donc partagée par l'ensemble des processeurs. L'image à Niveaux de Gris est uniquement accédée en lecture et pourra donc être dupliquée au niveau de chaque processeur.
- répartition uniforme des pixels de la File d'Attente courante sur n processeurs : il y a une FAH par processeur. Une phase d'initialisation répartit les pixels appartenant aux marqueurs initiaux de manière uniforme sur l'ensemble des processeurs. Lors du processus de propagation, chaque processeur répartit uniformément les pixels à empiler sur les autres processeurs. Pour la même raison que précédemment, l'image LABEL devra être partagée par l'ensemble des processeurs et l'image à Niveaux de Gris est dupliquée au niveau de chaque processeur.

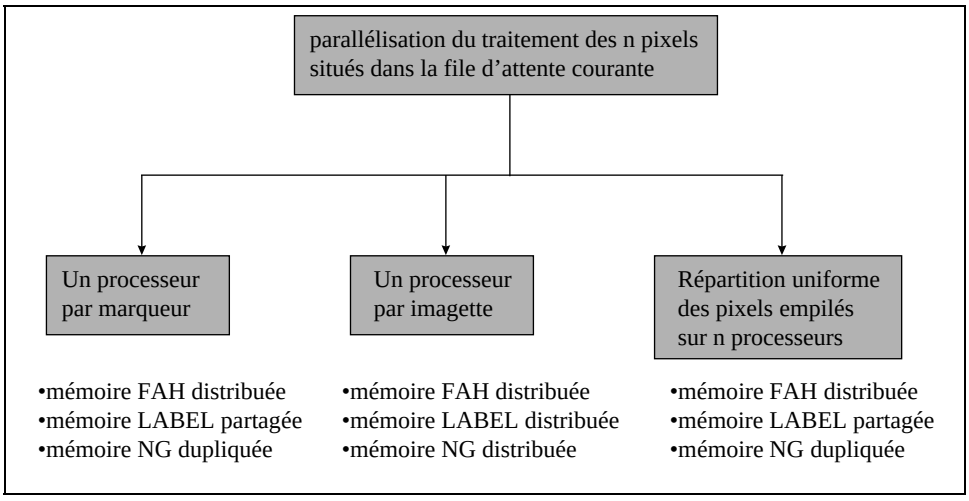


Figure 22 : Les différentes parallélisations envisagées

Quelque soit la solution adoptée, il faut résoudre les trois problèmes suivants :
- synchronisation inter-processeur sur le niveau de gris des pixels traités.
- synchronisation inter-processeur sur la distance géodésique pour un niveau de gris donné.
- échanges des pixels des bords dans le cas « un processeur par imagette » ou dans le cas « Répartition uniforme des pixels empilés sur n processeurs ».

Les échanges de pixels sont spécifiques à chaque architecture. En revanche, le contrôle inter-processeur en vue de respecter la double hiérarchie est commun aux trois architectures. Voyons dès maintenant en quoi il consiste.

4.2.1.2 Contrôles inter-processeur

Nous avons vu que la nécessité de respecter la hiérarchie des niveaux de gris et la distance géodésique à l'intérieur des plateaux impose un contrôle global sur l'ensemble des processeurs.
Un processeur ne peut passer au niveau de gris suivant que si :
- sa file d'attente courante est vide
- les files d'attentes courantes de tous les autres processeurs sont également vides.

Soit x le nombre de pixels situés dans la file d'attente courante au moment où le processeur commence à dépiler son premier pixel. La hiérarchie liée à la distance géodésique impose qu'un processeur ne traite que les x premiers pixels de la file d'attente même si d'autres pixels sont venus dans cette même file d'attente. On dira alors que le processeur est inactif bien que sa file d'attente courante ne soit pas vide. Une fois dans cet état, un processeur ne peut redevenir actif et traiter les pixels suivant que si tous les autres processeurs sont également inactifs. Ce processus se répète jusqu'à ce que les files d'attente courantes de tous les processeurs soient vides. On passe alors au niveau de gris suivant.

Nous avons tout d'abord envisagé d'effectuer ces deux contrôles au moyen d'un contrôleur centralisé. Ce contrôleur reçoit l'état d'activité de tous les processeurs

ainsi que l'état de leur file d'attente courante. Suivant l'état des processeurs, il renvoie à chaque processeur un signal de contrôle les faisant passer de l'état inactif à l'état actif. Le contrôleur a alors une structure qui dépend du nombre de processeurs ce qui n'est pas en accord avec une architecture évolutive en fonction de ce nombre de processeurs . C'est la raison pour laquelle nous avons également étudié le cas d'un contrôle distribué. Ces deux solutions sont présentées dans le chapitre 5.3 dans le cas de la parallélisation « un processeur par imagette ».

4.2.2 Parallélisation du cycle de traitement d'un pixel

Les opérations à effectuer pour traiter un pixel sont les suivantes :

- dépiler un pixel de la mémoire FAH
- lire la mémoire LABEL pour connaître le label du pixel dépilé

Pour chaque voisin du pixel dépilé :

- lire la mémoire LABEL
- Si le pixel voisin n'est pas étiqueté alors :
  - écrire dans la mémoire LABEL
  - empiler dans la mémoire FAH

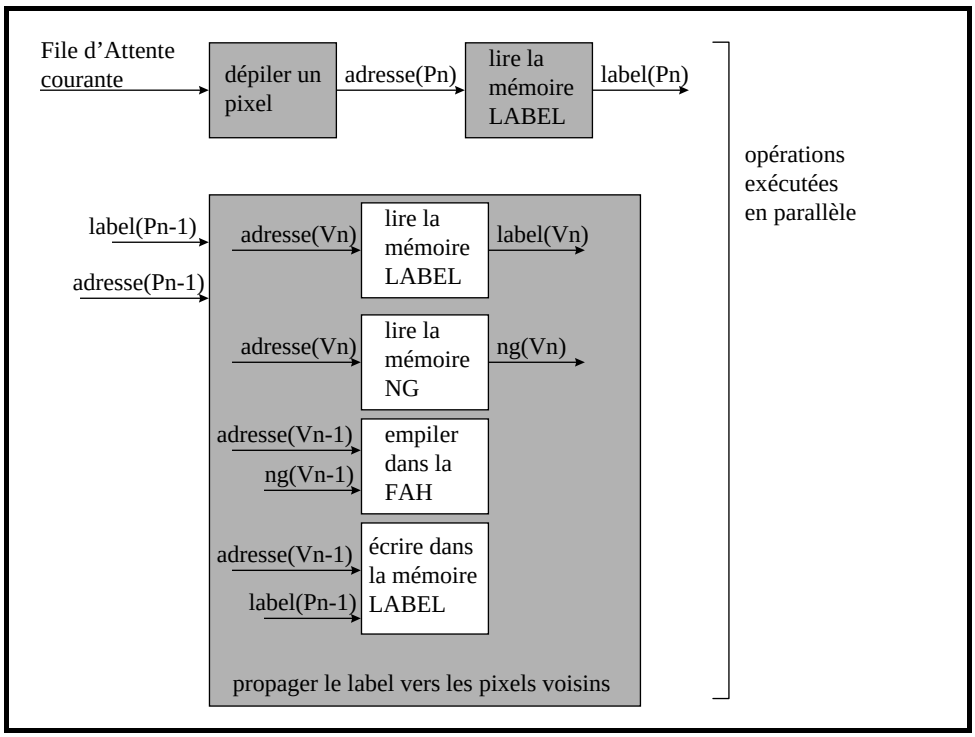


Figure 23 : parallélisation du traitement d'un pixel

Les n pixels situés dans la file d'attente courante au début de son traitement peuvent être traité simultanément. Nous proposons donc d'effectuer deux traitements en parallèle:

- dépiler un pixel (Pn) et lire son label (label(Pn)).

- propager le label du pixel dépilé avant Pn (Pn-1) vers son voisinage. Cette étape peut elle même être parallélisée en deux étapes :
    - lire le label et le niveau de gris d'un voisin Vn : label(Vn) et ng(Vn)
    - écrire le label de Pn-1 à l'adresse du voisin Vn-1 et empiler V(n-1), Vn-1 étant le voisin dont le label et le niveau de gris ont été lus avant le voisin Vn.

Ainsi, lorsque le traitement du voisinage du pixel Pn-1 est terminé, on peut aussitôt commencer à traiter le voisinage du pixel Pn de la File d'Attente courante.

4.3 Un processeur par imagettes

4.3.1 Introduction

Le modèle de parallélisation présenté ci-dessous est un modèle de parallélisme spatial : l'image est divisée en un nombre de blocs (appelés également imagettes) égal au nombre de processeurs. Ainsi, chaque processeur exécute l'algorithme de ZPE par FAH sur son propre bloc.

4.3.2 Représentation de cette solution

Dans cette représentation, nous avons choisi le cas d'un partage de l'image en horizontal et en vertical. Ainsi chaque processeur traite une imagette de 2K/N colonnes par 2L/N lignes, K et L étant respectivement le nombre de colonnes et de lignes de l'image de départ et N le nombre de processeurs (9 dans notre exemple). Cette parallélisation est représentée à la Figure 24.

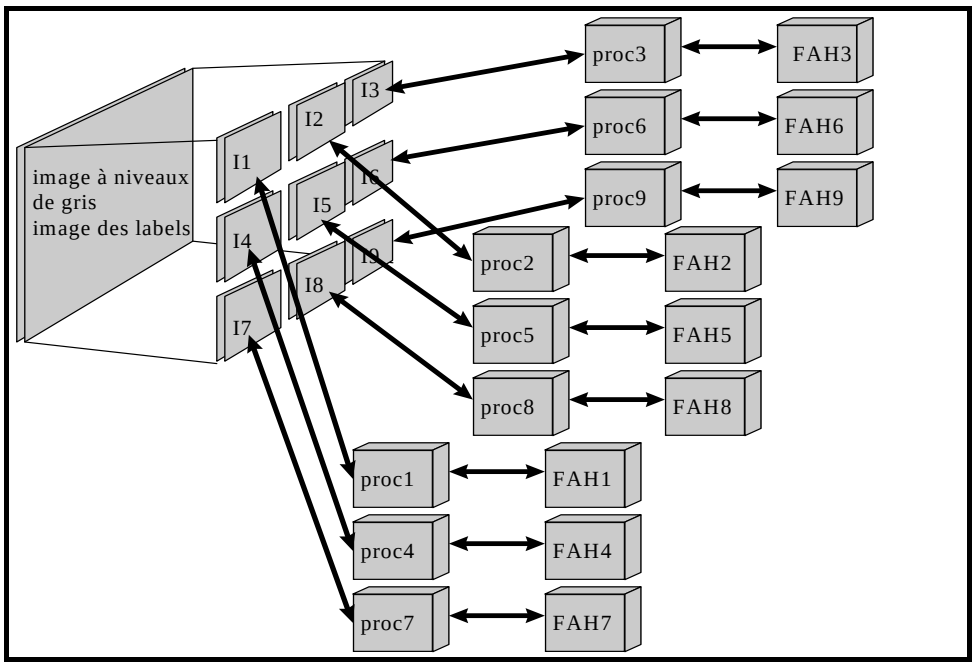


Figure 24 : parallélisation en 9 imagettes

Dans cette parallélisation, les mémoires de l'image LABEL, de l'image à niveau de gris et de la FAH sont distribuées sur les N processeurs. On a alors, pour chaque processeur, l'architecture représentée à la Figure 25. La propagation des marqueurs consiste à propager le label du pixel dépilé vers ses pixels voisins non encore étiquetés. Dans le cas où le pixel dépilé appartient au bord de l'imagette, certains pixels voisins n'appartiennent pas à la même zone mémoire que le pixel dépilé. Il faudra donc échanger des données entre processeurs pour propager le marqueur entre les pixels du bord des imagettes : ceci est matérialisé par le bus de données inter-processeur dans la Figure 25. D'autre part, le contrôle global assurant que les processeurs traitent tous le même niveau de gris ou la même distance à l'intérieur d'un niveau de gris donné, est représenté par le bus de contrôle inter-processeur dans la Figure 25.

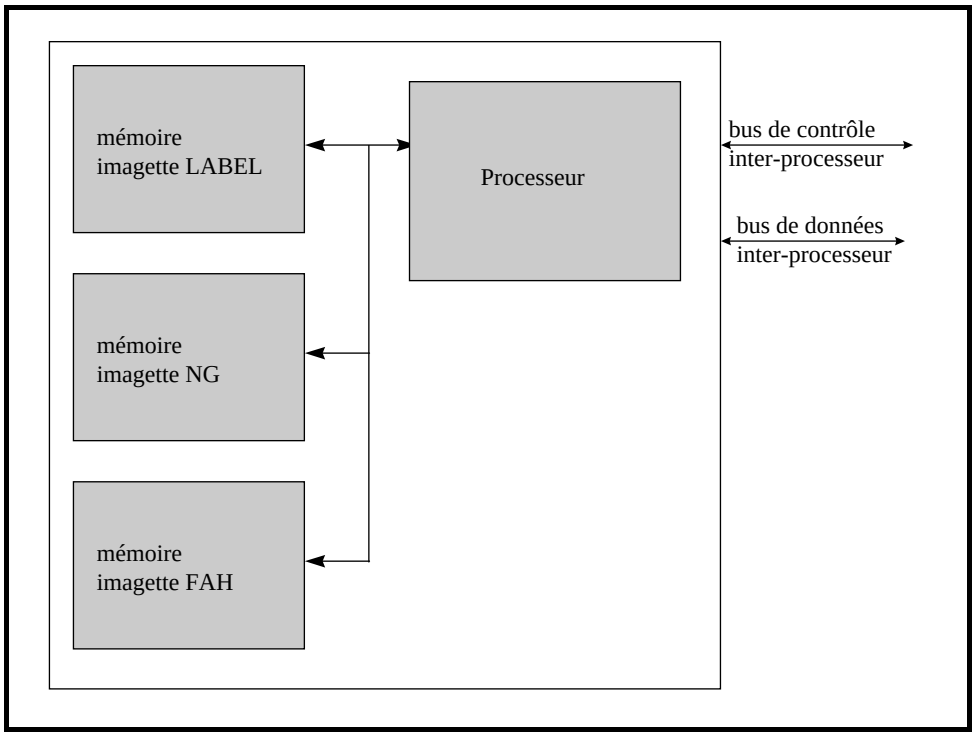


Figure 25 : Interconnection d'un processeur

4.3.3 Exemple de propagation des marqueurs

Dans cette section, nous proposons un exemple de propagation des marqueurs avec 9 processeurs et un partage de l'image selon une matrice carrée. Les images de départ sont représentées dans la Figure 26. Les processeurs ont été numérotés de 1 à 9 de la gauche vers la droite, et de haut en bas. L'étape d'initialisation et les premières étapes de propagation sont données en annexe. Sur deux colonnes, nous avons représenté pour chaque processeur l'état de leur activité et l'état de leur File d'Attente. L'adresse des pixels dans chaque imagette a été représentée de la manière suivante : ligne, colonne. Voici quelques commentaires sur chacune des étapes :

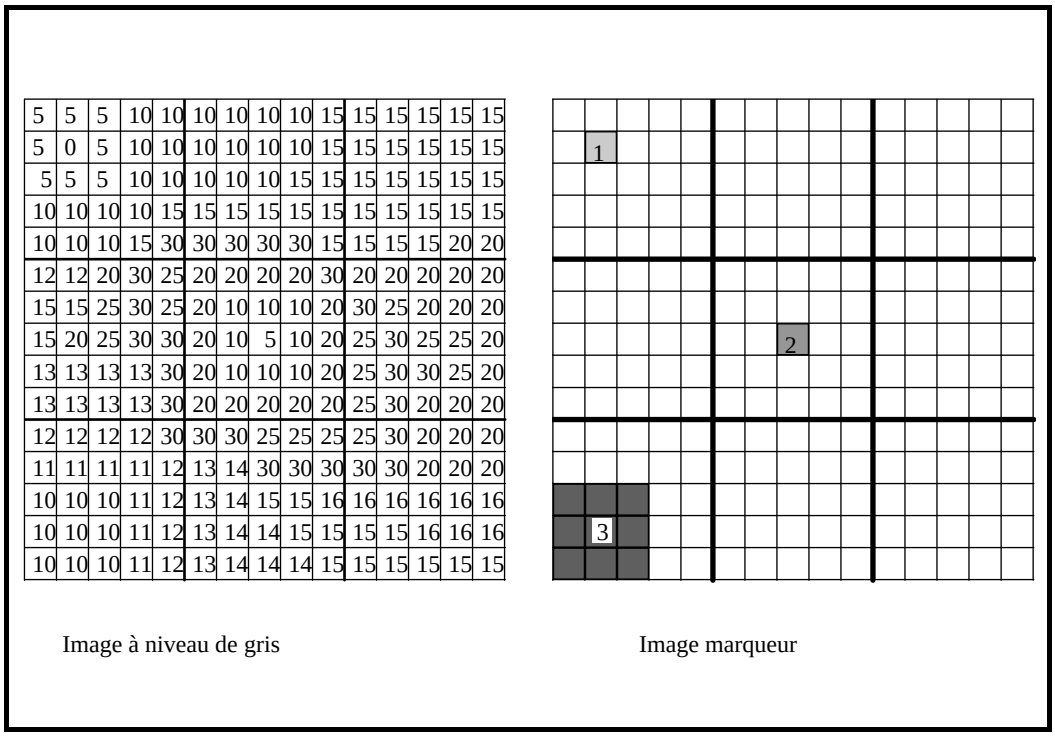


Figure 26 : image à niveaux de gris et image marqueur de l'exemple

Initialisation

chaque processeur stocke dans sa FAH les adresses des pixels déjà marqués et voisins de pixels non marqués. Seules les imagettes 1, 5 et 7 ont des marqueurs donc seuls les processeurs 1, 5 et 7 stockent des pixels dans leur FAH.

Propagation : étape 1

ce sont les pixels de niveau de gris 0 qui sont traités. Seul le processeur 1 est actif. Il dépile le pixel d'adresse (2,2), propage le marqueur 1 vers tous les voisins et les empile dans la FAH.

Propagation : étape 2

il n'y a pas de niveau de gris entre 0 et 5. Durant cette étape, c'est donc le niveau de gris 5 qui est traité par les processeurs 1 et 5. Ce sont des pixels de niveau de gris 10 qui sont étiquetés et empilés dans les Files d'Attente (FA).

Propagation : étape 3

il n'y a pas de niveau de gris entre 5 et 10. Ce sont donc les pixels de niveau de gris 10 qui sont maintenant traités. Trois processeurs sont simultanément actifs : 1, 5 et 7.

Propagation : étape 4

c'est toujours les pixels de niveau de gris 10 qui sont traités. En effet, durant l'étape 3, ce sont les pixels de niveau de gris 10 et situés à une distance 1 du bord descendant qui étaient dépilés. Maintenant, ce sont les pixels situés à une distance 2 du bord descendant. Cette étape illustre la nécessité d'un échange entre processeur car il faut pouvoir propager le marqueur des pixels appartenant aux bords des imagettes vers les pixels voisins n'appartenant pas à la même imagette. Prenons par exemple le cas

du pixel d'adresse (1,5) dans l'imagette 1. Lorsque ce pixel est dépilé, son marqueur doit être propagé vers les pixels d'adresses (1,1) et (2,1) de l'imagette 2. Il faut également empiler les adresses de ces pixels dans la FAH du processeur 2 afin qu'il puisse à son tour propager le marqueur 1. Nous verrons dans la suite de ce chapitre les solutions qui s'offrent à nous et plus particulièrement celle que nous avons choisie.

Propagation : étape 5
les pixels de niveau de gris 10 empilés dans la FAH du processeur 2 ont déclenché son activité. Il propage alors à son tour le marqueur 1 sur le plateau de niveau de gris 10.

Les étapes suivantes suivent le même principe que ces 5 premières étapes. Nous ne les avons donc pas décrites.

4.3.4 Choix du partage de l'image

Quel type de découpage faut-il envisager? horizontal, vertical ou en imagettes?

Pour partager l'image nous avons deux solutions :
- selon une matrice carrée.
- en bandes horizontales ou verticales.

Ces deux types de partage sont représentés dans la Figure 27.

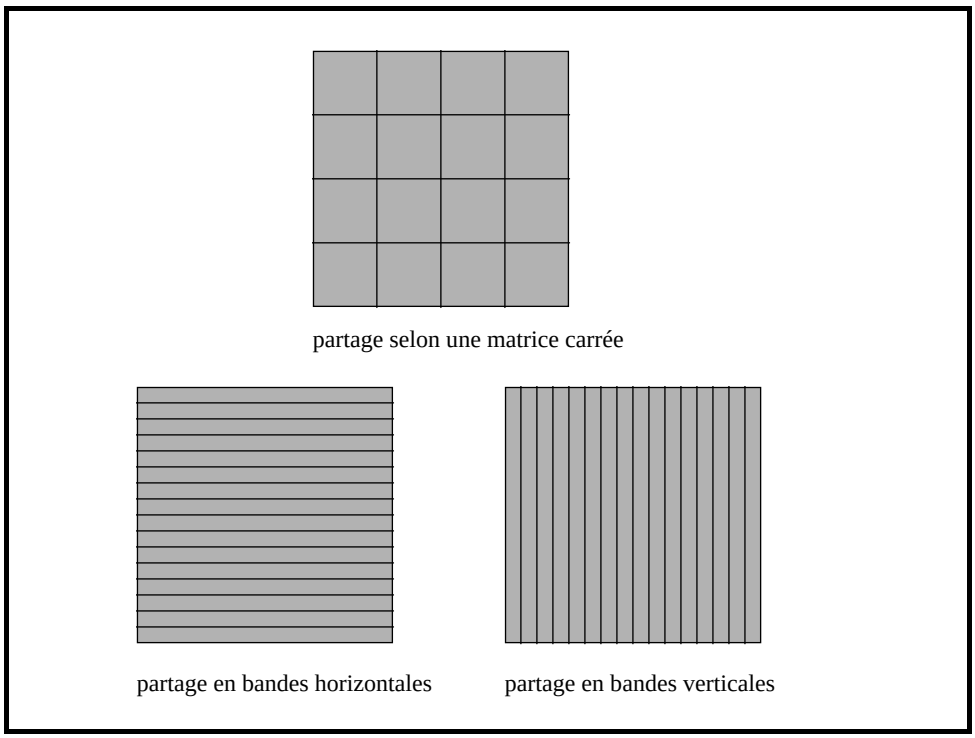


Figure 27 : types de partage des images pour un même nombre de processeurs

La différence essentielle entre ces deux types de partage est la longueur des bords en contact avec d'autres imagettes. Soit L et K le nombre de lignes et de colonnes de l'image de départ et N^2 le nombre de processeurs. Dans le cas général (imagette non située au bord de l'image globale), le nombre de pixels appartenant aux bords d'une imagette est :

- (2*(K+L)/N) + 4 dans le cas de la matrice carrée
- 2*K ou 2*L dans le cas respectivement du partage en bandes horizontales ou verticales.

Pour un nombre de processeurs supérieur à quatre, le partage de l'image selon une matrice carrée est préférable en ce qui concerne la quantité d'échanges de pixels. Dans la suite de ce rapport, nous avons considéré le cas du partage selon une matrice carrée car il est ensuite très facile de se ramener au cas du partage en bandes.

4.3.5 Traitement des pixels appartenant aux bords des imagettes

Reprenons le processus de traitement d'un pixel voisin V du pixel dépilé (pc) :

- obtention du label de V pour savoir s'il est déjà étiqueté ou non ;
- dans le cas où V n'est pas encore étiqueté alors :
    - étiquetage de V par le label de pc.
    - empilage de l'adresse de V dans la FAH.

Cela signifie que pour traiter V, dans le cas où V n'est pas encore étiqueté, il faut accéder deux fois à la mémoire image LABEL (en lecture puis en écriture), une fois la mémoire image NG et une fois la mémoire FAH.

Dans le cas où V n'appartient pas à la même imagette que pc, il est donc préférable que V soit traité par le processeur associé à V à partir d'une requête faite par le processeur qui a dépilé pc.

Notons P(V) le processeur traitant l'imagette contenant V. Pour que P(V) puisse étiqueter V et l'empiler dans sa FAH il faut qu'il connaisse :

- l'adresse de V
- le label de pc

Avant de déterminer les informations à envoyer au processeur récepteur, considérons les différents cas d'échange de pixels entre processeurs illustrés dans la Figure 28. Dans la Figure 28.a, nous avons représenté le cas des pixels situés aux angles des imagettes. Dans ce cas, le processeur traitant *pc* doit envoyer une requête à ces trois processeurs voisins pour qu'ils étiquettent les pixels les concernant et adjacents à *pc.* La majorité des cas est représenté par la Figure 28.b. Le processeur traitant *pc* doit faire une requête à son processeur voisin pour qu'il traite les 3 pixels voisins de *pc* le concernant.

Afin de réduire les échanges entre processeurs et éviter d'envoyer au processeur récepteur les adresses des pixels voisins de *pc*, c'est l'adresse de *pc* qui est envoyée et c'est au processeur récepteur de calculer les adresses des pixels voisins de *pc* qu'il

doit étiqueter et empiler dans sa FAH. Ainsi les données envoyées dans une requête sont :

- l'adresse du pixel dépilé (*pc*)
- le label de *pc.*

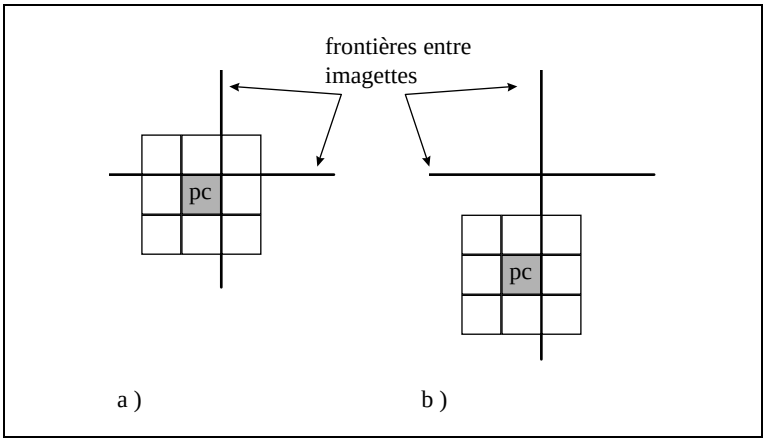


Figure 28 : deux cas d'échange de pixels entre processeurs

4.3.6 Gestion de l'échange des pixels

Nous avons envisagé deux solutions pour effectuer les requêtes entre processeur :

- par interruption.
- selon le principe de la boîte aux lettres.

La solution basée sur des interruptions a deux inconvénients :

- lorsqu'un processeur envoie une requête, il doit attendre d'avoir l'acquittement du processeur récepteur pour effectuer sa requête. Cela implique des délais supplémentaires sur les temps de traitement.
- dans le cas d'un partage de l'image selon une matrice carrée, il existe des configurations dans lesquelles les processeurs vont s'attendre les uns les autres ce qui aboutit à un « dead lock ». Cette situation est représentée dans la Figure 29. Dans ce cas, les processeurs vont s'attendre les uns les autres. Notons que ce type de bouclage n'existe pas dans le cas d'un partage de l'image en bandes.

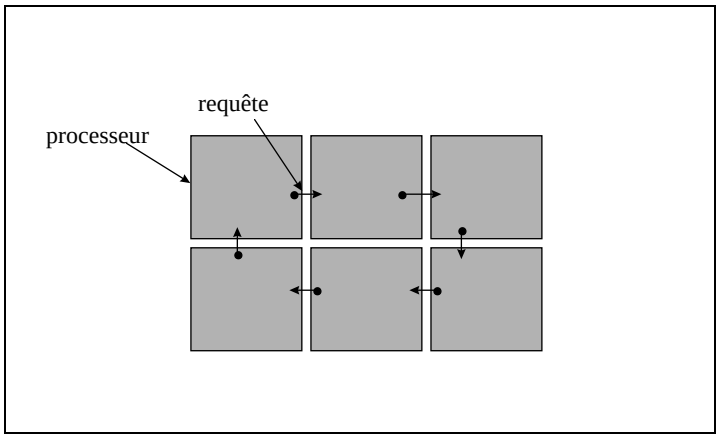


Figure 29 : configuration de requête aboutissant à un blocage

Nous avons donc décidé de baser l'échange de pixels sur le principe de la boite aux lettres. Cela signifie que les pixels émis par les processeurs voisins sont stockés dans une zone tampon en attendant d'être traités. Chaque processeur doit regarder régulièrement si sa zone tampon n'est pas vide. Si c'est le cas, il doit traiter les pixels reçus jusqu'à ce que la zone tampon soit vide.
Le moment où les pixels reçus doivent être traités par rapport au pixel de la file d'attente locale n'a pas d'importance. En effet, on a vu que les pixels d'un même niveau de gris où à la même distance du bord descendant dans un niveau de gris donné peuvent être traités dans n'importe quel ordre. On a donc le choix de les traiter :

- après avoir dépilé où empilé un pixel de la FA courante ;
- après avoir dépilé un pixel et traité tous ses pixels voisins ;
- au moment où le processeur a vidé sa FA courante.

Néanmoins, plus on privilégie le traitement des pixels de la FA locale par rapport aux pixels reçus, plus la taille de la zone tampon doit être importante.

4.3.7 Perspectives

Dans le cas d'un partage selon une matrice carrée, les processeurs qui traitent les imagettes non situées au bord de l'image globale échangent des pixels avec 8 processeurs voisins comme ceci est illustré dans la Figure 30. Cependant, comme nous l'avons montré dans la Figure 28.a, les liens diagonaux entre processeurs sont utilisés pour échanger seulement deux pixels entre les deux processeurs.

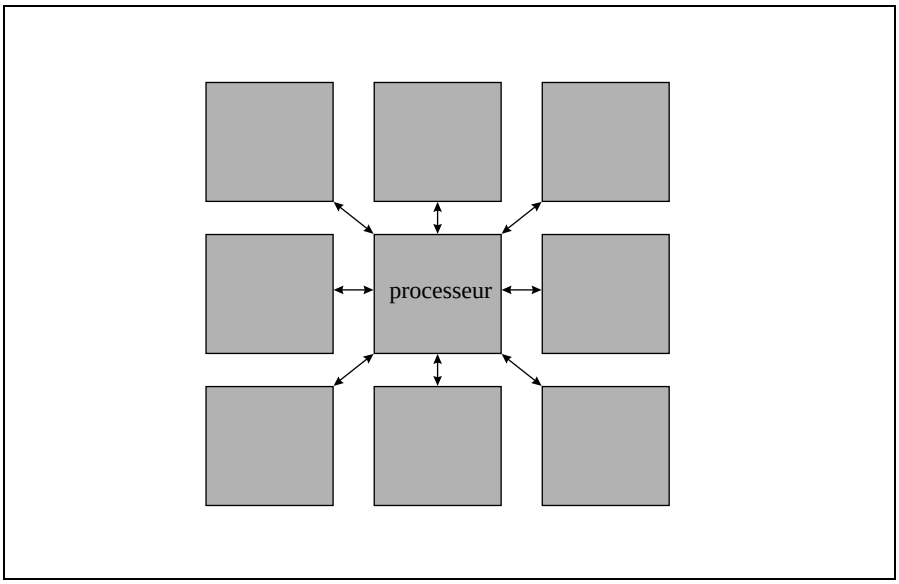


Figure 30 : maille d'interconnexion des processeurs

Pour éviter d'avoir un lien pour échanger seulement deux pixels, nous avons envisagé d'étudier une architecture utilisant seulement 4 liens inter-processeur. Il faudra alors déterminer une méthode pour échanger les pixels situés aux angles des imagettes au travers des 4 liens restants.

4.3.8 Contraintes

Afin d'être assuré que cette architecture soit utilisée au maximum de ces performances, il faut que les processeurs soient actifs en même temps le plus souvent possible. Pour cela on doit avoir équi-répartition des marqueurs sur l'ensemble de l'image.

4.4 Un processeur par marqueur

4.4.1 Présentation

Un schéma de cette architecture est donnée dans la Figure 31.

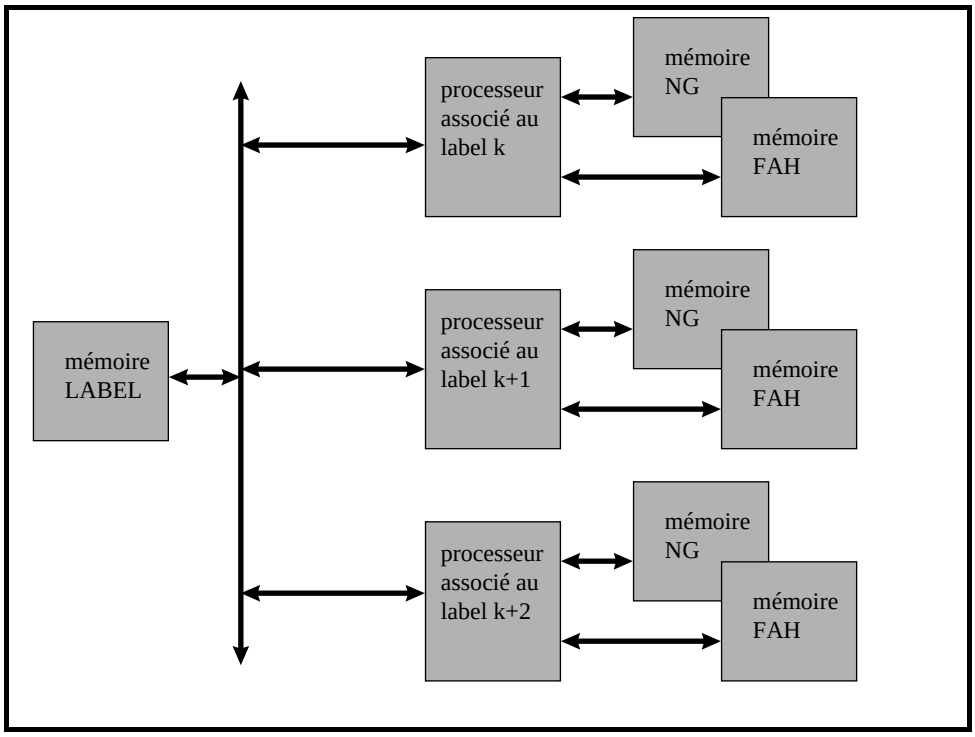


Figure 31 : architecture issue de la parallélisation par marqueur

La mémoire LABEL est partagée entre les processeurs car :

- chaque processeur n'a pas d'espace d'adressage pré-défini comme dans la solution « un processeur par imagette »
- chaque processeur lit et écrit dans cette image.

La mémoire FAH est distribuée au niveau de chaque processeur. Néanmoins, ne connaissant pas a priori la surface occupée par chaque marqueur, les FAH locales devront avoir une taille équivalente à celle nécessaire pour une architecture monoprocesseur. Dans chacune d'elle sont stockés des pixels de même label. L'image à niveau de gris étant uniquement accédée en lecture, elle peut être dupliquée au niveau de chaque processeur.

Cette répartition des pixels suivant leur label permet de simplifier l'algorithme de ZPE de départ. Celui-ci est alors le suivant :

- création de la FAH associée au label k : elle contient autant de piles qu'il y a de niveaux de gris dans l'image à segmenter
- initialisation de la FAH associée au label k : on empile dans la FAH les pixels appartenant aux marqueurs de label k qui sont voisins de pixels non marqués.
- gestion de la FAH (processus de propagation) : elle consiste à itérer les deux étapes suivantes jusqu'à ce que la FAH soit vide :
    - on dépile le premier pixel de la file d'attente courante. Appelons ce pixel « P_0 ».

- on empile dans la File d'Attente correspondant à leur niveau de gris tous les pixels voisins de P_0 qui n'ont pas déjà été étiquetés et on leur affecte le label k.

4.4.2 Communication inter-processeur

L'image LABEL étant partagée entre les processeurs, il n'est plus nécessaire d'échanger des labels entre processeurs. Il n'y a donc pas de temps perdu dans la communication inter-processeur.

4.4.3 Contrainte

L'inconvénient de cette architecture est que ses performances sont dépendantes du nombre de marqueurs. On doit donc déterminer une image marqueur avec le plus de marqueurs possibles et au maximum un nombre égal au nombre de processeurs.

4.5 Répartition uniforme des pixels à empiler sur n processeurs

Le schéma cette architecture est représenté dans la Figure 32. Il s'agit de répartir les pixels à empiler uniformément dans la FAH locale à chaque processeur.

Le traitement des pixels est alors exécuté en deux phases :

- 1ère phase : en parallèle, chaque processeur dépile les pixels situés dans sa file d'attente courante et envoie sur le bus d'échange les pixels à empiler sous la forme de jetons. Un jeton est constitué de l'adresse du pixel, de son label et du numéro du processeur destinataire. Le numéro du processeur destinataire est incrémenté à chaque fois qu'un jeton est émis ce qui permet de répartir uniformément les pixels sur les FAH locales. Chaque processeur reçoit les jetons dans une boîte aux lettres.

- 2ème phase : une fois que les processeurs ont vidé leur file d'attente, ils empilent dans leur FAH locale les jetons situés dans leur boîte aux lettres.

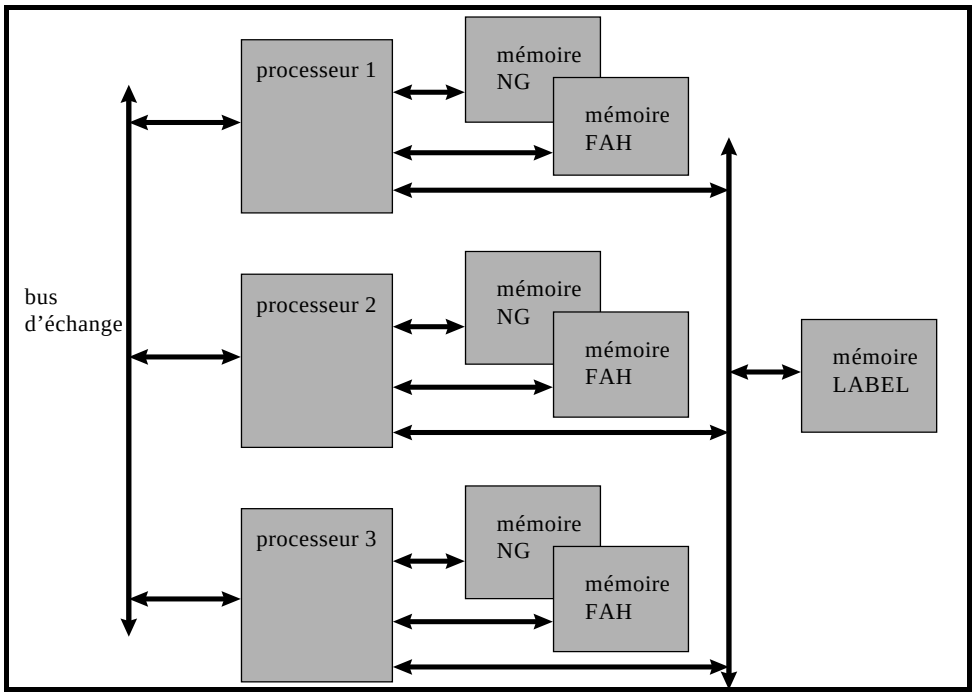


Figure 32 : un processeur par pixel de la File d'Attente courante

En ce qui concerne le bus d'échange, nous proposons de se baser sur le principe des anneaux de processeurs tel que celui illustré dans la Figure 33 : chaque processeur reçoit des jetons venant des processeurs situés à sa gauche. Chaque processeur émet vers les processeurs situés à sa droite les jetons qui venaient de sa gauche et qui ne lui étaient pas destinés ainsi que ses propres jetons.

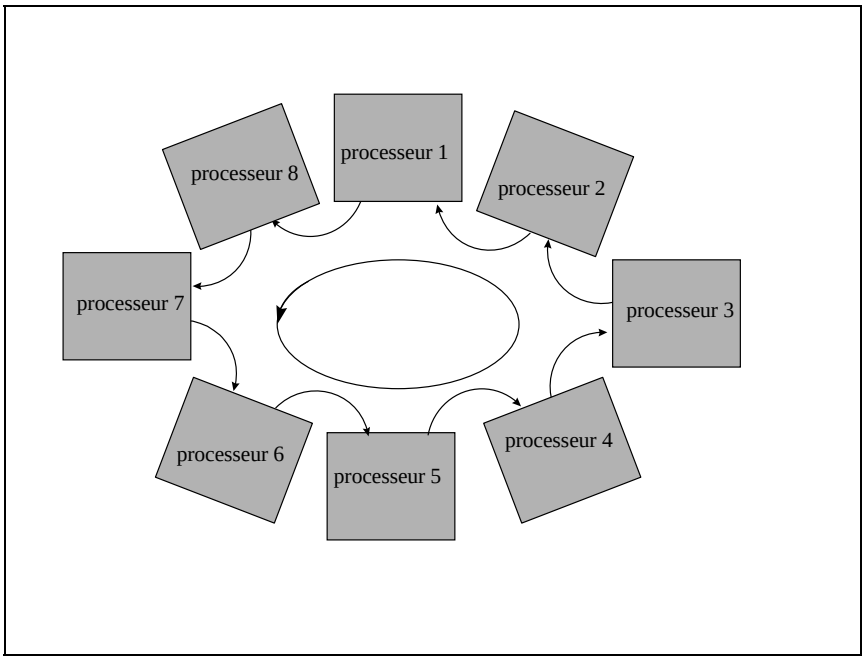


Figure 33 : anneau de processeurs

Cette architecture permet de répartir uniformément la charge de travail concernant l'empilement et le dépilement quel que soit le type d'image. Il reste à résoudre le problème de contention de bus au niveau de l'accès à la mémoire LABEL.

4.6 Conclusion

La solution consistant à partager l'image de départ en imagette est la seule qui permette de distribuer complètement la mémoire sur les processeurs. C'est la raison qui a motivé le décision d'étudier plus spécifiquement l'algorithme sous-jacent à cette architecture.

5. Parallélisation par imagettes : traitement parallèle distribué

5.1 Introduction

L'objet de ce chapitre est de décrire sous forme d'algorithmes un traitement parallèle distribué du processus d'inondation par File d'Attente Hiérarchique. Chaque processeur traite une file d'attente hiérarchique locale pour segmenter un sous-ensemble de l'image.
Ce chapitre est organisé selon le plan suivant: dans la section 1 nous présentons l'algorithme de traitement exécuté par chaque processeur. Les sections 2 et 3 décrivent les algorithmes de synchronisation et de communication. L'algorithme d'initialisation des files d'attente hiérarchiques locales est décrit dans la section 4. L'algorithme complet est décrit dans la section 5. Pour l'ensemble de ces algorithmes nous considérons le cas général d'une matrice [I][J] de processeurs . Dans la suite on notera par P_{ij} , $i \in [0..I-1]$, $j \in [0..J-1]$ un processeur de la matrice [I][J] .

5.2 Algorithme de traitement

P	position d'un pixel dans une image
V(p)	voisinage 3 × 3 de p
$X_{ij}(p)$	niveau de gris de p dans la sous image à segmenter X_{ij}
$Y_{ij}(p)$	label de p dans la sous image des labels Y_{ij}
$in\_HQ_{ij}(x, y)$	empiler x dans la file d'attente de priorité y du processeur P_{ij}
$out\_HQ_{ij}(y)$	dépiler de la file d'attente de priorité y du processeur P_{ij}
$level\_HQ_{ij}(y)$	nombre d'éléments dans la file d'attente de priorité y du processeur P_{ij}
Lm_{ij}	espace d'adressage du processeur P_{ij}
$queue\_empty_{ij}$	indicateur positionné à true par le processeur P_{ij} lorsque la file d'attente qu'il a finie de traiter est vide

Tableau 1 : notations

Le traitement exécuté par chaque processeur sur sa file d'attente locale est décrit par l'algorithme 1. Les notations utilisées sont précisées dans le Tableau 1.

La procédure send_process est détaillée dans la section 4.

```
traitement:  nb_dep <= level_HQi j(current_priority );
             for i in 1 to nb_dep do
                    p <= out_HQi j(current_priority);
                    level_HQi j(current_priority) <= level_HQi j(current_priority)  - 1;
                    for each p' ∈ (V(p) ∩ Lmi j)
                           if (Yi j(p') = 0) then
                                  Yi j(p') <= Y(p);
                                  in_HQi j(p', max(current_priority,Xi j(p'));
                                  level_HQi j(Xi j(p'))  <= level_HQi j(Xi j(p')) + 1;
                           end if;
                    end for;
                    call send_process(p);
             end for;
             if (level_HQi j(current_priority) = 0 ) then
                    queue_empty_ij <= true;
             else
                    queue_empty_ij <= false;
             end if;
             current_priority  <= current_priority + 1 ;
```

algorithme 1 : traitement

5.3 Synchronisation

Nous avons vu que le problème posé par le traitement parallèle de la FAH était celui de la synchronisation des processeurs. Autrement dit, lorsque les processeurs ont terminé de traiter leurs files d'attente locales de priorité *k* il faut pouvoir déterminer cette terminaison pour réactiver tous les processeurs afin qu'ils traitent leur file d'attente de priorité *k+1* ou ne réactiver qu'un sous-ensemble de processeurs pour qui les files d'attente locales précédemment traitées ne sont pas vides.

On notera par $ready_{ij}$ un indicateur d'état positionné à true par chaque processeur P_{ij} qui a terminé de traiter une file d'attente. On notera également par $start\_same_{ij}$ et $start\_next_{ij}$ deux indicateurs autorisant un processeur P_{ij} à traiter soit la file d'attente précédemment traitée soit la file d'attente de priorité égale à la priorité de la file

d'attente précédemment traitée plus un. Les équations qui régissent ces deux indicateurs sont :

$$start\_next_{ij} = (\cap ready_{ij}) \cap (\cap queue\_empty_{ij})$$
$$start\_same_{ij} = (\cap ready_{ij}) \cap (\cup \neg queue\_empty_{ij})$$

L'algorithme 2 décrit l'utilisation de ces indicateurs.

```
synchro:    ready_ij <= true;
            wait until (start_next_ij or start_same_ij = true)
                  ready_ij <= false;
                  if (start_next_ij = true) then
                        call traitement;
                  else
                        current_priority <= current_priority - 1;
                        call traitement;
                  end if;
            end wait;
            goto synchro;
```

algorithme 2 : synchronisation

Les deux types de contrôle que l'on peut adopter pour générer les signaux $start\_next_{ij}$ et $start\_same_{ij}$ sont décrits dans les paragraphes ci-dessous.

5.3.1 Contrôle global centralisé

Ce contrôle est illustré par la Figure 34. Le contrôleur global reçoit les signaux $ready_{ij}$ et $queue\_empty_{ij}$. Il renvoie à tous les processeurs les signaux *start_next* et *start_same*. L'algorithme du contrôleur est le suivant:

```
ctrl_gb: start_next <= false;
         start_same <= false;
         wait until (∩ready_ij = true)
               if (∩queue_empty_ij = true) then
                     start_next <= true;
               else
                     start_same <= true;
               end if;
         end wait;
         goto ctrl_gb;
```

algorithme 3 : contrôleur

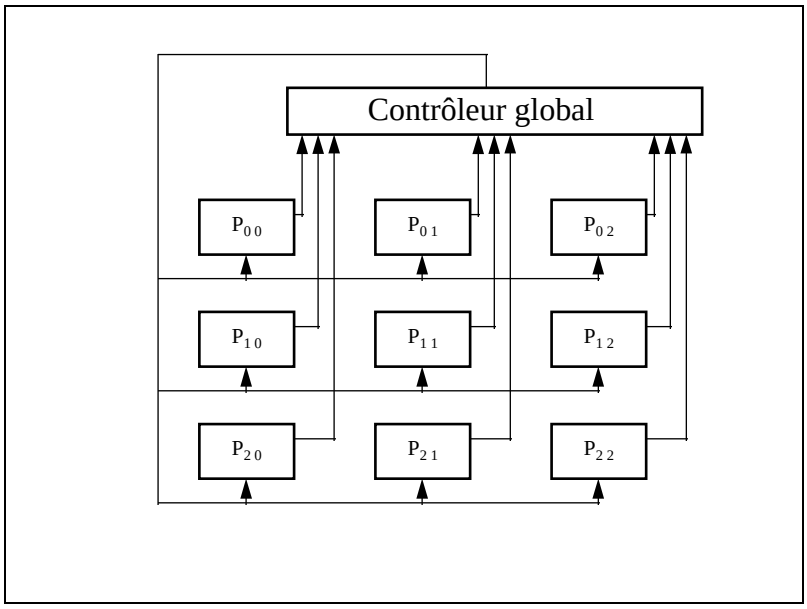


Figure 34 : Contrôle centralisé

5.3.2 Contrôle global distribué

Ce contrôle est illustré par la Figure 35. Les signaux *start_next* et *start_same* sont générés par le processeur $P_{0\,0}$. Ce dernier comporte un module de contrôle qui exécute l'algorithme suivant:

```
ctrl_dt: start_next <= false;
         start_same <= false;
         wait until (ready_I-1 J-1 = true)
                 if (queue_empty_I-1 J-1 = true) then
                         start_next <= true;
                 else
                         start_same <= true;
                 end if;
         end wait;
         goto ctrl_dt;
```

algorithme 4 : contrôle distribué

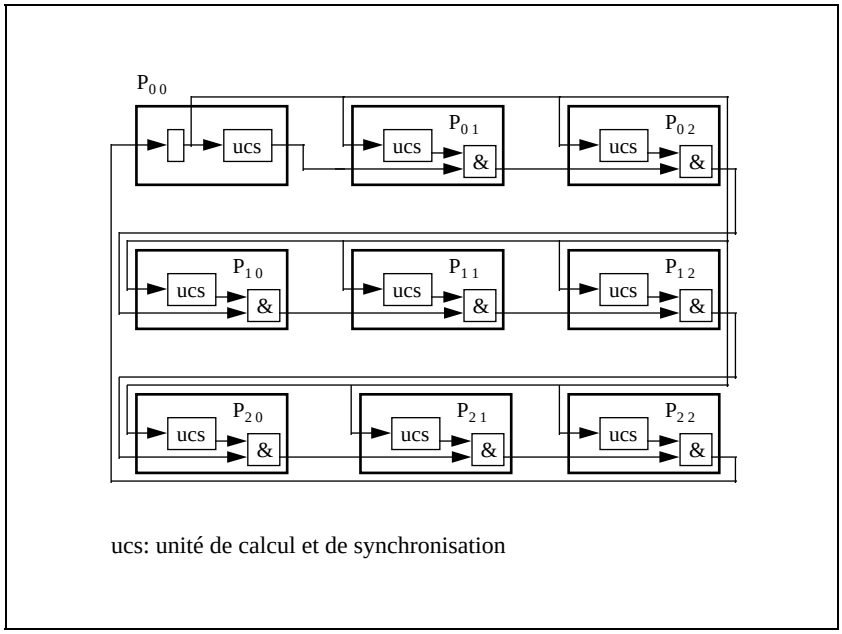


Figure 35 : Contrôle distribué

5.4 Communication

Quand un ou plusieurs processeurs dépilent des pixels appartenant au bord de leurs sous-images, ils doivent communiquer la position de ces pixels ainsi que leurs labels aux processeurs ayant une frontière commune. Les procédures de communication sont décrites par la Figure 3. Chaque processeur communique avec chacun de ces huit voisins au moyen de deux tampons: un tampon d'émission et un tampon de réception. Les tampons pour les communications horizontales et verticales ont des tailles égales aux longueurs des bords des sous-images. Pour les communications diagonales les tampons ont une taille unitaire.

Les notations utilisées pour les communications sont données dans Tableau 2 :

send(tampon_$s_{k\,l}$, x, y)	mettre les données x et y dans le tampon d'émission k l
receive(tampon_$r_{k\,l}$, x, y)	réceptionner les données x et y du tampon de réception k l
tampon_$full_{k\,l}$	indicateur positionné à true si le tampon de réception k l n'est pas vide

Tableau 2 : notations pour les communications

Les indices k et l sont relatifs au processeur $P_{i\,j}$:

$(k,l) \in \{(i,j+1), (i,j-1), (i+1,j), (i-1,j), (i+1,j+1), (i+1,j-1), (i-1,j+1), (i-1, j-1)\}$

```
send_process:     for each p' ∈ V(p) / V(p) ∩ Lm_ij
                        for each (k,l)
                                if (p' ∈ Lm_kl)
                                        send(tampon_s_kl, p, Y(p));
                                end if;
                        end for;
                  end for;

receive_process:  while(tampon_full_kl = true)
                        receive(tampon_r_kl, x, y);
                        for each p' ∈ (V(x) ∩ Lm_ij)
                                if (Y_ij(p') = 0) then
                                        Y_ij(p') <= y;
                                        in_HQ_ij(p', X_ij(p'));
                                        level_HQ_ij(X_ij(p'))  <= level_HQ_ij(X_ij(p')) + 1;
                                end if;
                        end for;
                  end while;
                  if (level_HQ_ij(current_priority) = 0 ) then
                        queue_empty_ij <= true;
                  else
                        queue_empty_ij <= false;
                  end if;
                  current_priority  <= current_priority + 1;
```

algorithme 5 : procédures de communication

5.5 L'algorithme complet

L'algorithme 6 décrit l'initialisation des files d'attente. On notera par $init_{ij}$ un indicateur autorisant un processeur P_{ij} à initialiser ses files d'attentes locales. L'indicateur end_init_{ij} est positionné à true lorsque le processeur a terminé son étape d'initialisation.

On notera par M et N les longueurs des cotés des imagettes.

L'algorithme 7 regroupe l'ensemble des algorithmes précédents.

```
ini_process: end_init_ij <= false;
             wait until (init_ij = true)
                  for p in [i .. i+M-1] × [j .. j+N-1] do
                       init_hq <= false;
                       if (Y(p) ≠ 0) then
                            for each p' ∈ V(p)
                                 if (Y(p') = 0) then
                                      init_hq <= true;
                                 end if;
                            end for;
                       end if;
                       if (init_hq = true) then
                            in_HQij(p, Xij(p));
                            level_HQij(Xij(p'))  <= level_HQij(Xij(p')) + 1;
                       end if;
                  end for;
             end wait;
             end_init_ij <= true;
```

algorithme 6 : initialisation

```
             call ini_process;
hq_process:  ready_ij <= true;
             while (start_next_ij or start_same_ij or ∪tampon_full_kl = true)
                  ready_ij <= false;
                  if (start_next_ij = true) then
                       call traitement;
                  else
                       if (start_same_ij = true)
                            current_priority <= current_priority - 1;
                            call traitement;
                       else
                            for each (k,l)
                                 if (tampon_full_kl = true)
                                      call receive_process(k,l);
                                 end if;
                            end for;
                       end if;
                  end if;
             end while;
             goto hq_process;
```

algorithme 7 : traitement global

6. Conclusion

L'étude algorithmique de la parallélisation de la LPE effectuée à partir de l'algorithme de file d'attente hiérarchique standard nous a permis de dégager les contraintes hiérarchiques que cette parallélisation doit absolument respecter sous peine de grèver fortement la qualité et l'exactitude du résultat. Elle a permis également de dégager les parties du processus où l'effort de parallélisation pouvait être entrepris. L'analyse détaillée des différentes approches possibles a montré que le découpage en imagettes comme l'approche de processeurs distribués par marqueurs pouvaient se montrer d'un rendement décevant, notamment lorsqu'on doit réaliser une LPE contrôlée par marqueurs, parce que dans ce cas, il y a à la fois peu de marqueurs et les zones à inonder donc les pixels de même priorité ont peu de chance de se retrouver équitablement répartis entre toutes les imagettes. L'approche distribuée semble donc la plus prometteuse même si elle est la plus complexe à mettre en oeuvre en ce qui concerne la gestion des FAH, de la mémoire image et de la mémoire-label. Il est probable cependant que l'augmentation de la vitesse de réalisation de la LPE nécessitera une combinaison de ces trois approches associée à une structure de type "pipe-line" au niveau de chaque processeur. L'approche imagette notamment a de l'intérêt ne serait-ce que parce qu'il faudra bien pour des raisons de coût et complexité de réalisation limiter la taille des images traitées. L'unique moyen de dépasser cette limitation sera alors de découper les grandes images en imagettes. Il faut donc, quelque soit l'architecture finalement retenue, être capable de gérer ce découpage et d'organiser les transferts d'informations entre processeurs pour réaliser une LPE exacte sur de telles images de grande taille.
La suite de l'étude sera consacrée à l'implantation de ces algorithmes dans l'environnement Ptolemy. Elle devra aider à finaliser l'architecture définitive et elle fournira des indications de performance et les gains de vitesse espérés en fonction du type de segmentation et du type d'images analysées.

7. Annexe

Cette annexe montre un exemple de propagation des marqueurs dans le cas d'un partage de l'image en neuf imagettes. A chaque étape est illustré l'état des FAH associées à chaque processeur. L'étape d'initialisation et les cinq premières étapes de propagation sont expliquées au paragraphe 4.3.3.

5	5	5	10	10	10	10	10	10	15	15	15	15	15	15
5	0	5	10	10	10	10	10	10	15	15	15	15	15	15
5	5	5	10	10	10	10	10	15	15	15	15	15	15	15
10	10	10	10	15	15	15	15	15	15	15	15	15	15	15
10	10	10	15	30	30	30	30	30	15	15	15	15	20	20
12	12	20	30	25	20	20	20	20	30	20	20	20	20	20
15	15	25	30	25	20	10	10	10	20	30	25	20	20	20
15	20	25	30	30	20	10	5	10	20	25	30	25	25	20
13	13	13	13	30	20	10	10	10	20	25	30	30	25	20
13	13	13	13	30	20	20	20	20	20	25	30	20	20	20
12	12	12	12	30	30	30	25	25	25	25	30	20	20	20
11	11	11	11	12	13	14	30	30	30	30	30	20	20	20
10	10	10	11	12	13	14	15	15	16	16	16	16	16	16
10	10	10	11	12	13	14	14	15	15	15	15	16	16	16
10	10	10	11	12	13	14	14	14	15	15	15	15	15	15

Image à niveau de gris

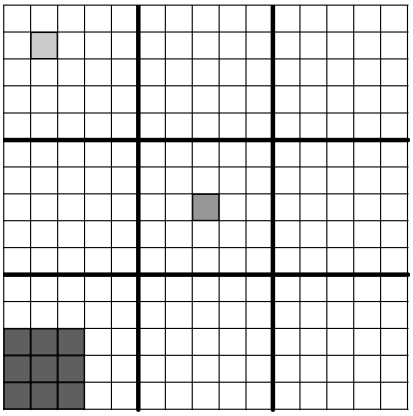
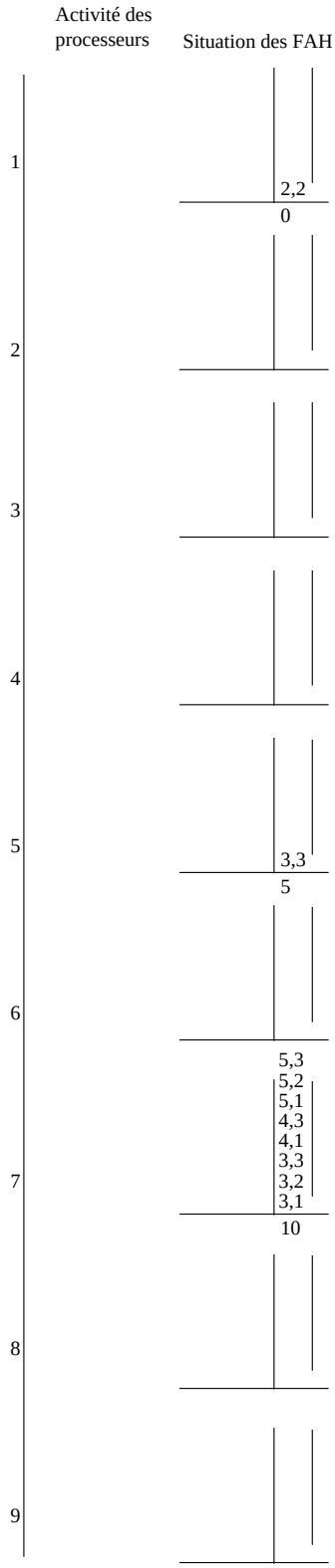


Image marqueur



Initialisation

5	5	5	10	10	10	10	10	10	15	15	15	15	15	15
5	0	5	10	10	10	10	10	10	15	15	15	15	15	15
5	5	5	10	10	10	10	10	15	15	15	15	15	15	15
10	10	10	10	15	15	15	15	15	15	15	15	15	15	15
10	10	10	15	30	30	30	30	30	15	15	15	15	20	20
12	12	20	30	25	20	20	20	20	30	20	20	20	20	20
15	15	25	30	25	20	10	10	10	20	30	25	20	20	20
15	20	25	30	30	20	10	5	10	20	25	30	25	25	20
13	13	13	13	30	20	10	10	10	20	25	30	30	25	20
13	13	13	13	30	20	20	20	20	20	25	30	20	20	20
12	12	12	12	30	30	30	25	25	25	25	30	20	20	20
11	11	11	11	12	13	14	30	30	30	30	30	20	20	20
10	10	10	11	12	13	14	15	15	16	16	16	16	16	16
10	10	10	11	12	13	14	14	15	15	15	15	16	16	16
10	10	10	11	12	13	14	14	14	15	15	15	15	15	15

Image à niveau de gris

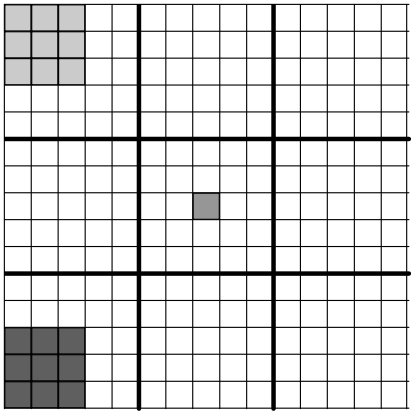
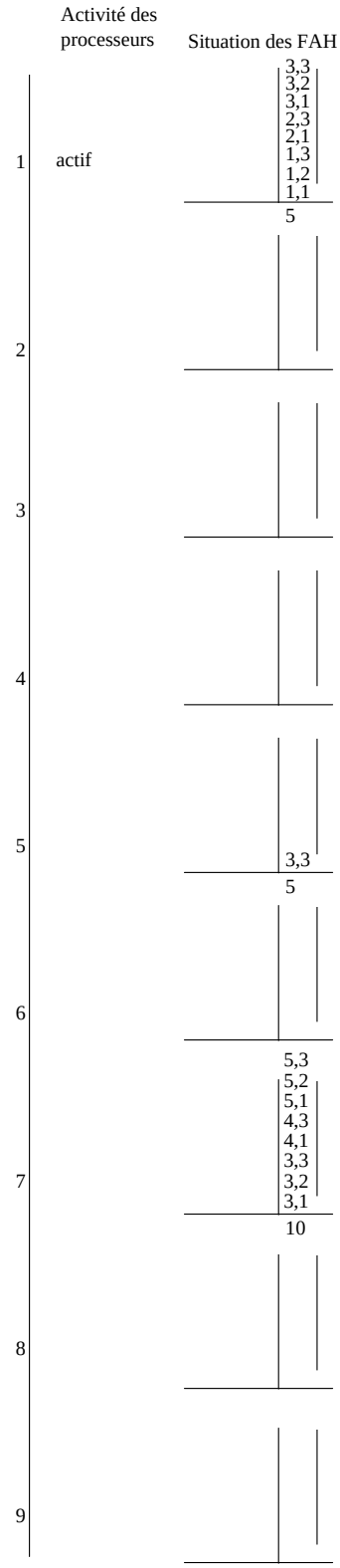


Image marqueur



Propagation : étape 1

5	5	5	10	10	10	10	10	10	15	15	15	15	15	15
5	0	5	10	10	10	10	10	10	15	15	15	15	15	15
5	5	5	10	10	10	10	10	15	15	15	15	15	15	15
10	10	10	10	15	15	15	15	15	15	15	15	15	15	15
10	10	10	15	30	30	30	30	30	15	15	15	15	20	20
12	12	20	30	25	20	20	20	20	30	20	20	20	20	20
15	15	25	30	25	20	10	10	10	20	30	25	20	20	20
15	20	25	30	30	20	10	5	10	20	25	30	25	25	20
13	13	13	13	30	20	10	10	10	20	25	30	30	25	20
13	13	13	13	30	20	20	20	20	20	25	30	20	20	20
12	12	12	12	30	30	30	25	25	25	25	30	20	20	20
11	11	11	11	12	13	14	30	30	30	30	30	20	20	20
10	10	10	11	12	13	14	15	15	16	16	16	16	16	16
10	10	10	11	12	13	14	14	15	15	15	15	16	16	16
10	10	10	11	12	13	14	14	14	15	15	15	15	15	15

Image à niveau de gris

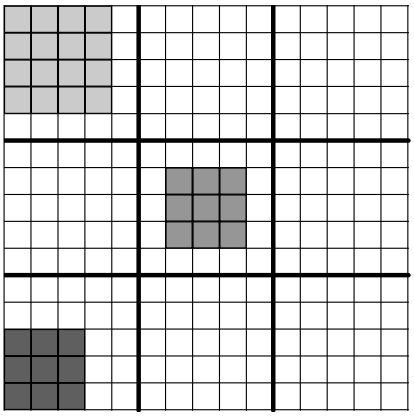
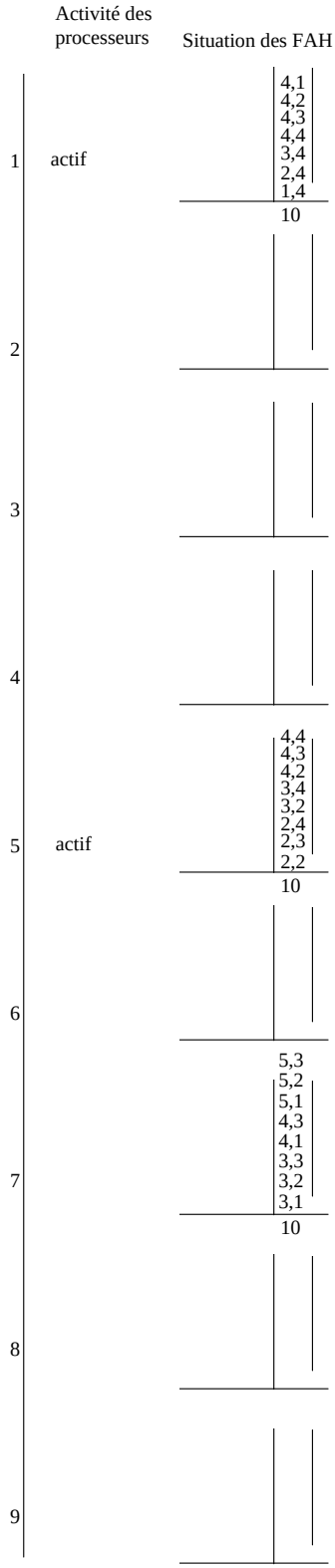


Image marqueur



Propagation : étape 2

5	5	5	10	10	10	10	10	10	15	15	15	15	15	15
5	0	5	10	10	10	10	10	10	15	15	15	15	15	15
5	5	5	10	10	10	10	10	15	15	15	15	15	15	15
10	10	10	10	15	15	15	15	15	15	15	15	15	15	15
10	10	10	15	30	30	30	30	30	15	15	15	15	20	20
12	12	20	30	25	20	20	20	20	30	20	20	20	20	20
15	15	25	30	25	20	10	10	10	20	30	25	20	20	20
15	20	25	30	30	20	10	5	10	20	25	30	25	25	20
13	13	13	13	30	20	10	10	10	20	25	30	30	25	20
13	13	13	13	30	20	20	20	20	20	25	30	20	20	20
12	12	12	12	30	30	30	25	25	25	25	30	20	20	20
11	11	11	11	12	13	14	30	30	30	30	30	20	20	20
10	10	10	11	12	13	14	15	15	16	16	16	16	16	16
10	10	10	11	12	13	14	14	15	15	15	15	16	16	16
10	10	10	11	12	13	14	14	14	15	15	15	15	15	15

Image à niveau de gris

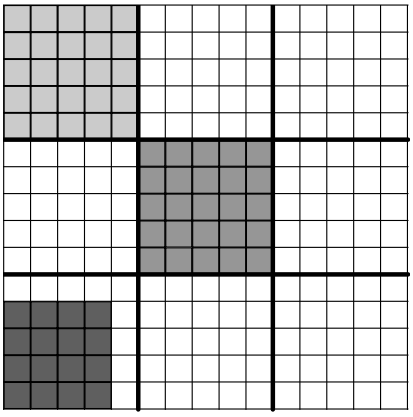
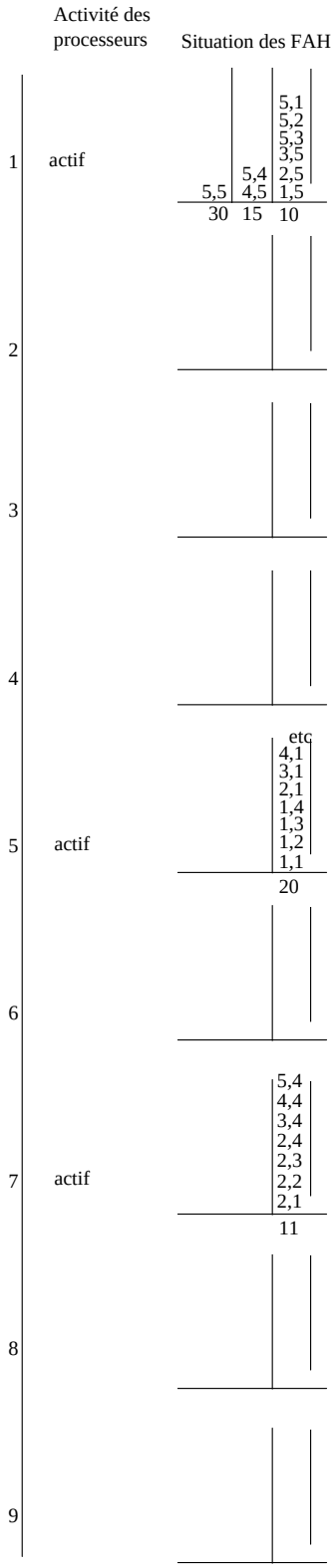


Image marqueur



Propagation : étape 3

5	5	5	10	10	10	10	10	10	15	15	15	15	15	15
5	0	5	10	10	10	10	10	10	15	15	15	15	15	15
5	5	5	10	10	10	10	10	10	15	15	15	15	15	15
10	10	10	10	15	15	15	15	15	15	15	15	15	15	15
10	10	10	15	30	30	30	30	30	15	15	15	15	20	20
12	12	20	30	25	20	20	20	20	30	20	20	20	20	20
15	15	25	30	25	20	10	10	10	20	30	25	20	20	20
15	20	25	30	30	20	10	5	10	20	25	30	25	25	20
13	13	13	13	30	20	10	10	10	20	25	30	30	25	20
13	13	13	13	30	20	20	20	20	20	25	30	20	20	20
12	12	12	12	30	30	30	25	25	25	25	30	20	20	20
11	11	11	11	12	13	14	30	30	30	30	30	20	20	20
10	10	10	11	12	13	14	15	15	16	16	16	16	16	16
10	10	10	11	12	13	14	14	15	15	15	15	16	16	16
10	10	10	11	12	13	14	14	14	15	15	15	15	15	15

Image à niveau de gris

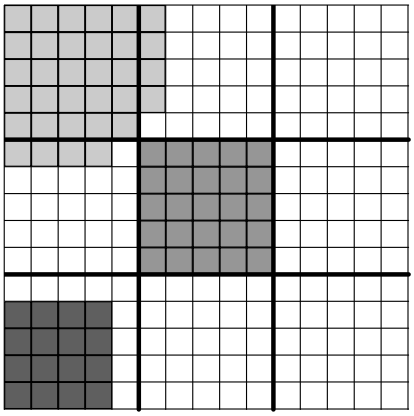
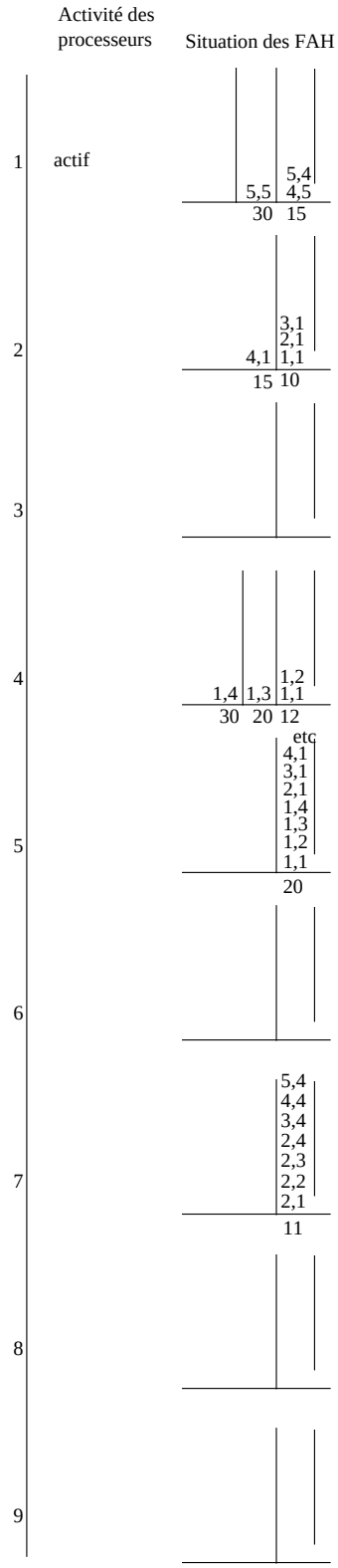


Image marqueur



Propagation : étape 4

5	5	5	10	10	10	10	10	10	15	15	15	15	15	15
5	0	5	10	10	10	10	10	10	15	15	15	15	15	15
5	5	5	10	10	10	10	10	15	15	15	15	15	15	15
10	10	10	10	15	15	15	15	15	15	15	15	15	15	15
10	10	10	15	30	30	30	30	30	15	15	15	15	20	20
12	12	20	30	25	20	20	20	20	30	20	20	20	20	20
15	15	25	30	25	20	10	10	10	20	30	25	20	20	20
15	20	25	30	30	20	10	5	10	20	25	30	25	25	20
13	13	13	13	30	20	10	10	10	20	25	30	30	25	20
13	13	13	13	30	20	20	20	20	20	25	30	20	20	20
12	12	12	12	30	30	30	25	25	25	25	30	20	20	20
11	11	11	11	12	13	14	30	30	30	30	30	20	20	20
10	10	10	11	12	13	14	15	15	16	16	16	16	16	16
10	10	10	11	12	13	14	14	15	15	15	15	16	16	16
10	10	10	11	12	13	14	14	14	15	15	15	15	15	15

Image à niveau de gris

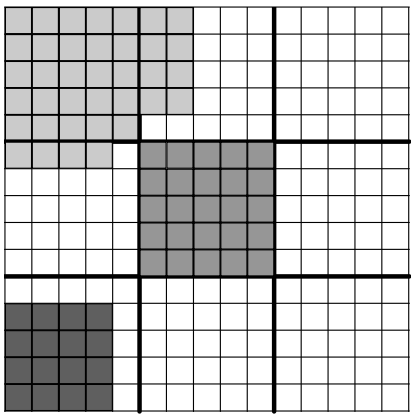
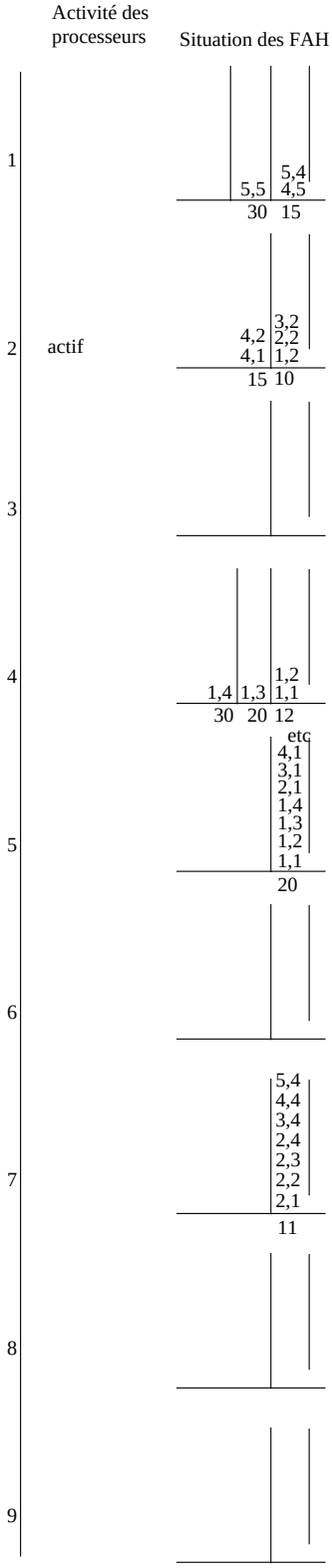


Image marqueur



Propagation : étape 5

5	5	5	10	10	10	10	10	10	15	15	15	15	15	15
5	0	5	10	10	10	10	10	10	15	15	15	15	15	15
5	5	5	10	10	10	10	10	15	15	15	15	15	15	15
10	10	10	10	15	15	15	15	15	15	15	15	15	15	15
10	10	10	15	30	30	30	30	30	15	15	15	15	20	20
12	12	20	30	25	20	20	20	20	30	20	20	20	20	20
15	15	25	30	25	20	10	10	10	20	30	25	20	20	20
15	20	25	30	30	20	10	5	10	20	25	30	25	25	20
13	13	13	13	30	20	10	10	10	20	25	30	30	25	20
13	13	13	13	30	20	20	20	20	20	25	30	20	20	20
12	12	12	12	30	30	30	25	25	25	25	30	20	20	20
11	11	11	11	12	13	14	30	30	30	30	30	20	20	20
10	10	10	11	12	13	14	15	15	16	16	16	16	16	16
10	10	10	11	12	13	14	14	15	15	15	15	16	16	16
10	10	10	11	12	13	14	14	14	15	15	15	15	15	15

Image à niveau de gris

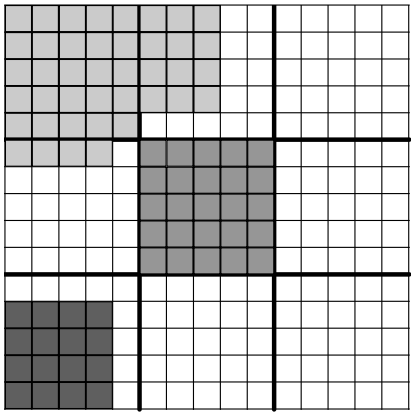
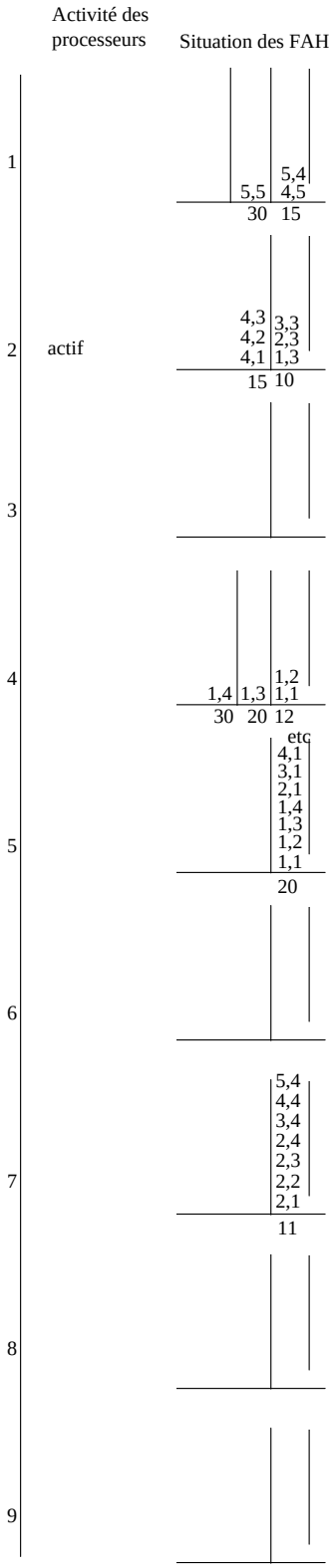


Image marqueur



Propagation : étape 6

5	5	5	10	10	10	10	10	10	15	15	15	15	15	15
5	0	5	10	10	10	10	10	10	15	15	15	15	15	15
5	5	5	10	10	10	10	10	15	15	15	15	15	15	15
10	10	10	10	15	15	15	15	15	15	15	15	15	15	15
10	10	10	15	30	30	30	30	30	15	15	15	15	20	20
12	12	20	30	25	20	20	20	20	30	20	20	20	20	20
15	15	25	30	25	20	10	10	10	20	30	25	20	20	20
15	20	25	30	30	20	10	5	10	20	25	30	25	25	20
13	13	13	13	30	20	10	10	10	20	25	30	30	25	20
13	13	13	13	30	20	20	20	20	20	25	30	20	20	20
12	12	12	12	30	30	30	25	25	25	25	30	20	20	20
11	11	11	11	12	13	14	30	30	30	30	30	20	20	20
10	10	10	11	12	13	14	15	15	16	16	16	16	16	16
10	10	10	11	12	13	14	14	15	15	15	15	16	16	16
10	10	10	11	12	13	14	14	14	15	15	15	15	15	15

Image à niveau de gris

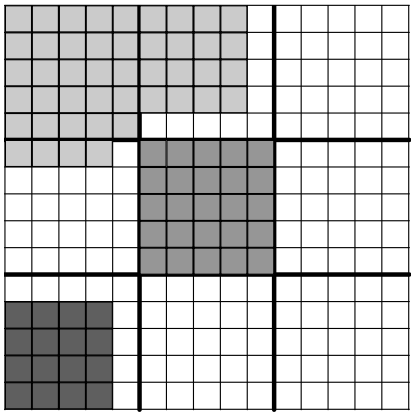
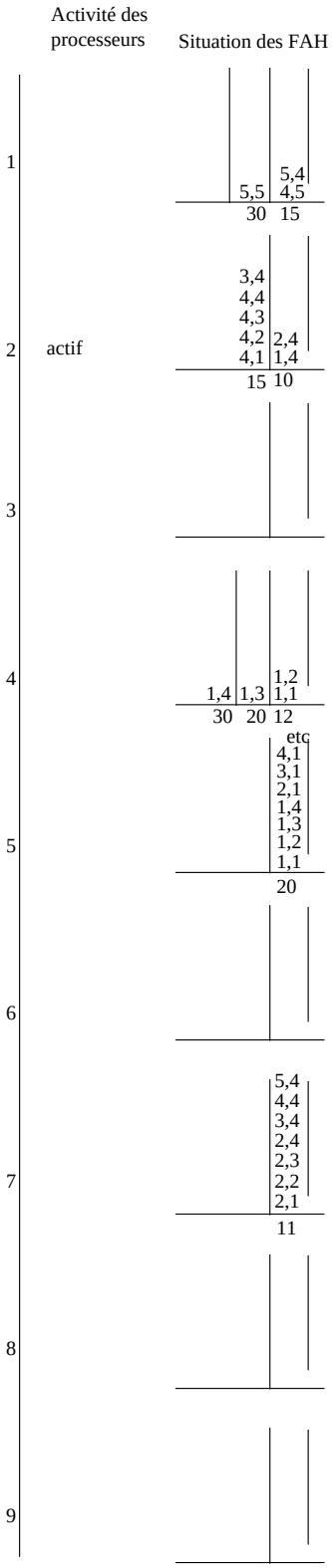


Image marqueur



Propagation : étape 7

5	5	5	10	10	10	10	10	10	15	15	15	15	15	15
5	0	5	10	10	10	10	10	10	15	15	15	15	15	15
5	5	5	10	10	10	10	10	15	15	15	15	15	15	15
10	10	10	10	15	15	15	15	15	15	15	15	15	15	15
10	10	10	15	30	30	30	30	30	15	15	15	15	20	20
12	12	20	30	25	20	20	20	20	30	20	20	20	20	20
15	15	25	30	25	20	10	10	10	20	30	25	20	20	20
15	20	25	30	30	20	10	5	10	20	25	30	25	25	20
13	13	13	13	30	20	10	10	10	20	25	30	30	25	20
13	13	13	13	30	20	20	20	20	20	25	30	20	20	20
12	12	12	12	30	30	30	25	25	25	25	30	20	20	20
11	11	11	11	12	13	14	30	30	30	30	30	20	20	20
10	10	10	11	12	13	14	15	15	16	16	16	16	16	16
10	10	10	11	12	13	14	14	15	15	15	15	16	16	16
10	10	10	11	12	13	14	14	14	15	15	15	15	15	15

Image à niveau de gris

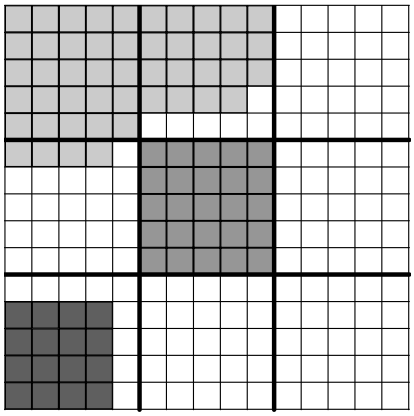
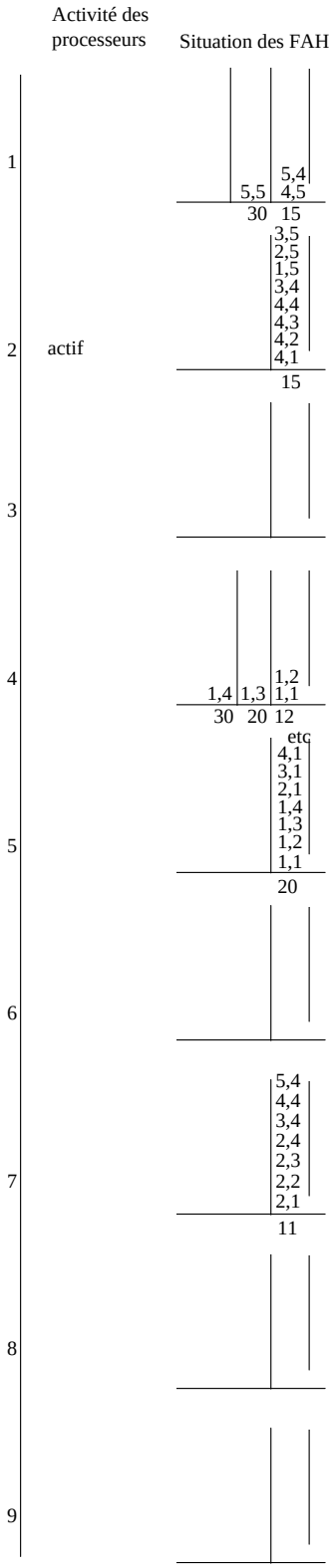


Image marqueur



Propagation : étape 8

5	5	5	10	10	10	10	10	10	15	15	15	15	15	15
5	0	5	10	10	10	10	10	10	15	15	15	15	15	15
5	5	5	10	10	10	10	10	15	15	15	15	15	15	15
10	10	10	10	15	15	15	15	15	15	15	15	15	15	15
10	10	10	15	30	30	30	30	30	15	15	15	15	20	20
12	12	20	30	25	20	20	20	20	30	20	20	20	20	20
15	15	25	30	25	20	10	10	10	20	30	25	20	20	20
15	20	25	30	30	20	10	5	10	20	25	30	25	25	20
13	13	13	13	30	20	10	10	10	20	25	30	30	25	20
13	13	13	13	30	20	20	20	20	20	25	30	20	20	20
12	12	12	12	30	30	30	25	25	25	25	30	20	20	20
11	11	11	11	12	13	14	30	30	30	30	30	20	20	20
10	10	10	11	12	13	14	15	15	16	16	16	16	16	16
10	10	10	11	12	13	14	14	15	15	15	15	16	16	16
10	10	10	11	12	13	14	14	14	15	15	15	15	15	15

Image à niveau de gris

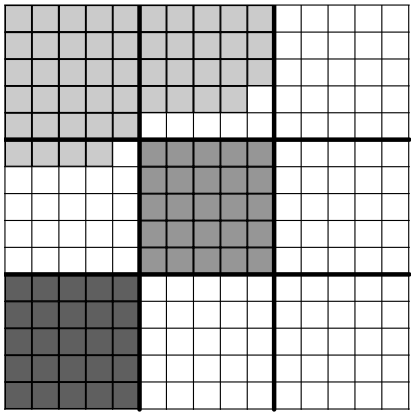
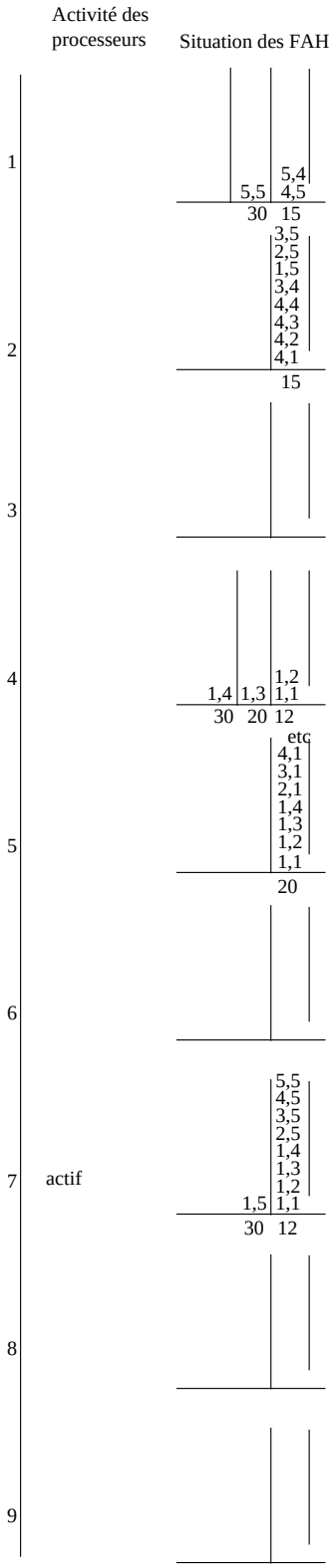


Image marqueur



Propagation : étape 9

5	5	5	10	10	10	10	10	10	15	15	15	15	15	15
5	0	5	10	10	10	10	10	10	15	15	15	15	15	15
5	5	5	10	10	10	10	10	15	15	15	15	15	15	15
10	10	10	10	15	15	15	15	15	15	15	15	15	15	15
10	10	10	15	30	30	30	30	30	15	15	15	15	20	20
12	12	20	30	25	20	20	20	20	30	20	20	20	20	20
15	15	25	30	25	20	10	10	10	20	30	25	20	20	20
15	20	25	30	30	20	10	5	10	20	25	30	25	25	20
13	13	13	13	30	20	10	10	10	20	25	30	30	25	20
13	13	13	13	30	20	20	20	20	20	25	30	20	20	20
12	12	12	12	30	30	30	25	25	25	25	30	20	20	20
11	11	11	11	12	13	14	30	30	30	30	30	20	20	20
10	10	10	11	12	13	14	15	15	16	16	16	16	16	16
10	10	10	11	12	13	14	14	15	15	15	15	16	16	16
10	10	10	11	12	13	14	14	14	15	15	15	15	15	15

Image à niveau de gris

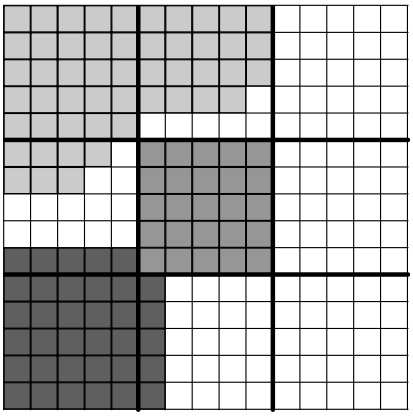
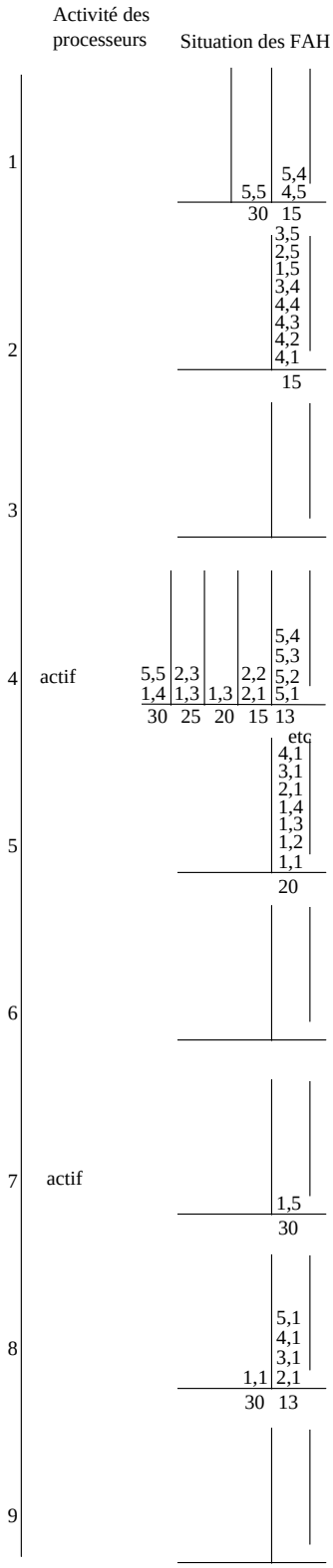


Image marqueur



Propagation : étape 10

5	5	5	10	10	10	10	10	10	15	15	15	15	15	15
5	0	5	10	10	10	10	10	10	15	15	15	15	15	15
5	5	5	10	10	10	10	10	15	15	15	15	15	15	15
10	10	10	10	15	15	15	15	15	15	15	15	15	15	15
10	10	10	15	30	30	30	30	30	15	15	15	15	20	20
12	12	20	30	25	20	20	20	20	30	20	20	20	20	20
15	15	25	30	25	20	10	10	10	20	30	25	20	20	20
15	20	25	30	30	20	10	5	10	20	25	30	25	25	20
13	13	13	13	30	20	10	10	10	20	25	30	30	25	20
13	13	13	13	30	20	20	20	20	20	25	30	20	20	20
12	12	12	12	30	30	30	25	25	25	25	30	20	20	20
11	11	11	11	12	13	14	30	30	30	30	30	20	20	20
10	10	10	11	12	13	14	15	15	16	16	16	16	16	16
10	10	10	11	12	13	14	14	15	15	15	15	16	16	16
10	10	10	11	12	13	14	14	14	15	15	15	15	15	15

Image à niveau de gris

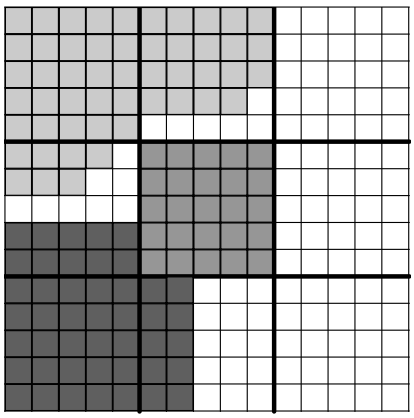
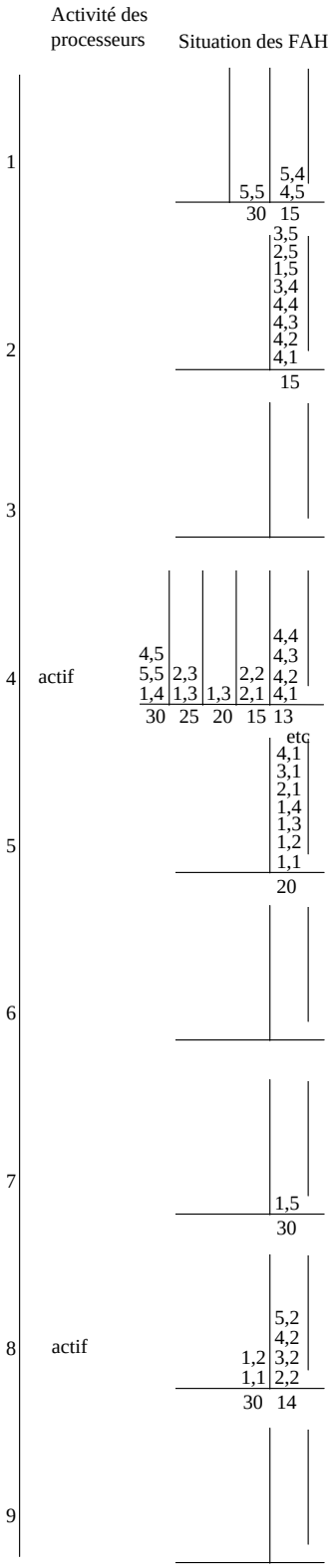


Image marqueur



Propagation : étape 11

5	5	5	10	10	10	10	10	10	15	15	15	15	15	15
5	0	5	10	10	10	10	10	10	15	15	15	15	15	15
5	5	5	10	10	10	10	10	15	15	15	15	15	15	15
10	10	10	10	15	15	15	15	15	15	15	15	15	15	15
10	10	10	15	30	30	30	30	30	15	15	15	15	20	20
12	12	20	30	25	20	20	20	20	30	20	20	20	20	20
15	15	25	30	25	20	10	10	10	20	30	25	20	20	20
15	20	25	30	30	20	10	5	10	20	25	30	25	25	20
13	13	13	13	30	20	10	10	10	20	25	30	30	25	20
13	13	13	13	30	20	20	20	20	20	25	30	20	20	20
12	12	12	12	30	30	30	25	25	25	25	30	20	20	20
11	11	11	11	12	13	14	30	30	30	30	30	20	20	20
10	10	10	11	12	13	14	15	15	16	16	16	16	16	16
10	10	10	11	12	13	14	14	15	15	15	15	16	16	16
10	10	10	11	12	13	14	14	14	15	15	15	15	15	15

Image à niveau de gris

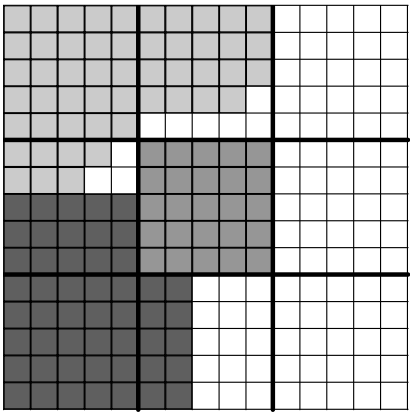
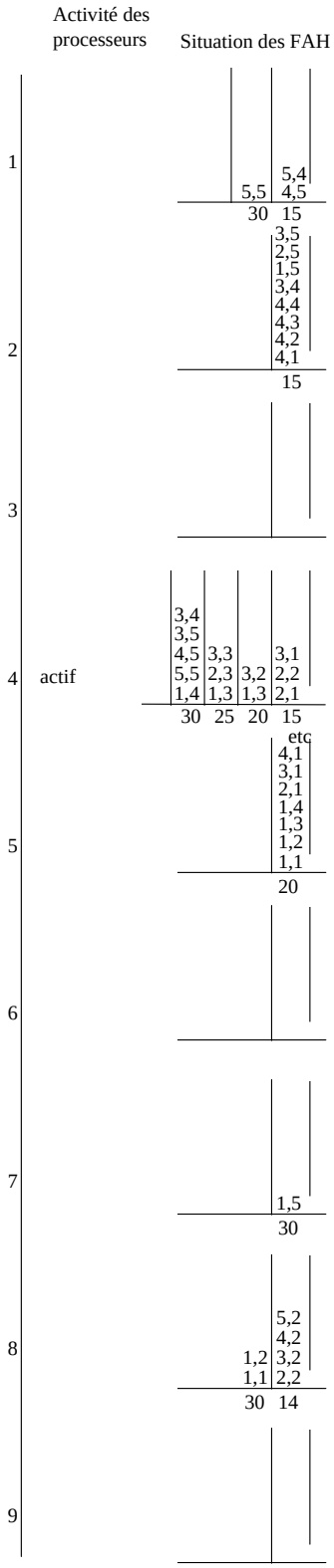


Image marqueur



Propagation : étape 12