Watersheds on edge or node weighted graphs

Fernand Meyer

Watersheds on edge or node weighted graphs

Fernand Meyer

CMM-Centre de Morphologie Mathématique,
Mathématiques et Systèmes, MINES ParisTech, France
fernand.meyer@mines-paristech.fr

Abstract. The paper proposes a unified watershed definition and algorithms for both node or edge weighted graphs. The flooding adjunction permits to associate to each type a common flooding graph in which the node and edge weights may be derived from each other. A lexicographic order relation between non ascending paths permits to order them according to their steepness. The watershed zones, i.e. the points linked with two distinct minima through non ascending paths are more and more reduced as one considers steeper paths and may be suppressed with .

1 Introduction

The watershed is a versatile and powerful segmentation tool. Its use for segmentation is due to Ch.Lantuéjoul and S.Beucher [6]. The history of its development is rich and the literature about it vast and confusing [43]:

- It may be applied on an image considered as a topographic surface, the pixels indicating the ground level [6], [36], [6],. Or it may be applied on a graph, such as a region adjacency graph [5], where the relevant information are the altitudes of the pass points between adjacent catchment basins . In the first case, one has to find the watershed on a node weighted graphs, in the second on an edge weighted graph [14].

- Some algorithms produce thin divide lines (or surfaces in 3D images) separating the catchment basins, others produce partition, where each region represents a catchment basin.

- Thinning algorithms modify the topography of the surface until flat zones appear in the catchment basins separated by divide lines. Other methods do not modifiy the relief

- Two physical models undeline the watershed concept. The rain model, where the destiny of a drop of water falling on the surface defines the catchment basins ; the divide line or zone is the set of nodes from where a drop of water may reach several distinct catchment basins. The flooding model where the relief is flooded from sources placed at the regional minima and meet along the divide lines. The later method being often implemented as shortest distance algorithms [39], [47].

A good review on the watershed may be found in [50], and in the recent book [62]

The present paper considers watersheds producing partitions. Its merit is to reconcile the rain model and the flooding model and to show that node and edge weighted graphs are perfectly equivalent and produce the same watershed partitions. The paper orders the trajectories of a drop of water or of a floodind according their steepness ; it shows also how the minimum spanning forests and minimum spanning trees may be ordered according this steepness.

The paper introduces the flooding adjunction and extracts from each weighted graph a flooding graph on which node and edge weights may be deduced from each other. A series of nested partial orders measuring the steepness of a trajectory of water are defined and a pruning algorithm presented for eliminating the trajectories with a low steepness.

The outline of the paper is the following

- a brief history of the watershed
- Reminders on adjunctions
- Reminders on weighted graphs
- Distances on node or edge weighted graphs
- Adjunctions between nodes and edges
- The flooding adjunction.
- Invariants of the flooding and closing adjunction
- Flooding graph
- Paths of steepest descent and k-steep graphs
- The scissor operator, minimum spanning forests and watershed partitions
- Lexicographic distances and SKIZ
- The waterfall hierarchy.
- Emergence and role of the minimum spanning tree
- Discussion and conclusion

Part I

The history of the watershed

2 Thinnings, geodesic distances and skeletons by zone of influence

The history of the watershed for segmentation is linked with the technological development of the image processing devices. In the mid seventies, computer memory was expensive, and computers slow. At the CMM we developed the first image analyzers, subsequently commercialized by Leitz under the name TAS holding binary image memories [26]. The result of an image transform may be stored in a memory and become the source of a second transform. Chaining operators permits new developments such as geodesic transforms, skeletons etc. The first watershed transform emerged from an alchemy mixing skeletons by zone of influence and binary thinning and thickening algorithms for constructing skeletons.

Christian Lantuejoul, in order to model a polycrystalline alloy, defined and studied the skeleton by zones of influence of a binary collection of grains in his thesis [27] ; he studied the geodesic metric used for constructing a SKIZ in [28]. However, at that time, the binary operators of the TAS did not permit to construct geodesic distances and SKIZ directly. He used instead binary homotopic thinnings for the construction of the SKIZ.

For studying the drainage properties of a topographic surface, he had the idea to construct geodesic SKIZs of the minima, taking as masks the successive thresholds of the function. This gave the first algorithm for the construction of watersheds. With Serge Beucher, they applied the watershed transform to the gradient image of gas bubbles, yielding the first watershed application to segmentation [6] [7].

The same method, applied to the more complex image of electrophoretic gels highlighted the major drawback of watershed segmentation : a severe oversegmentation, due to the presence of multiple spurious minima in the gradient image. I proposed a slight modification of the thinning algorithm which solved the problem. Instead of performing successive geodesic thickenings of all regional minima, one performs a thickening of a set of markers, some of them inside the objects to segment and at least one of them in the background [35],[4],[8]. This method produces a coarse approximation of the contours, between the inside and outside markers of the objects, as starting point of the successive geodesic homothetic thinnings. For increasing thresholds of the gradient image, the contours narrow down and ultimately produce the correct result. Marker driven watershed became the dominant morphological segmentation paradigm for some time [4].

Homotopic thinnings peel off points of a thick contour until this contour becomes thin, producing a thin line between the various markers or minima. G. Bertrand defined destructible points whose grey tone may be lowered without connecting adjacent catchment basins, yielding a kind of thinning for gray tone images. As a result he got what he called the topological watershed [3] where a thin line separates grey tone flat zones containing each a regional minimum of the initial surface and having the same grey tone as this minimum.

As a matter of fact, in terms of geodesic distance, one may be interested by the set of points equidistant from two distinct seeds, and obtain a skeleton by zone of influence, in form of a thin line. One may also be interested by the points which are closer to one seed than to any other seed. On a digital grid, there exist pairs of neighboring pixels, such that one is closer to a seed and the other closer to another seed, without a third pixel separating them. Other pixels are at the same distance of two seeds. For this reason, it is often preferred to create a tessellation, i.e. a partition of the image, where each tile is made of all pixels closer to a seed than to any other, but also contains some pixels which are equidistant from two seeds. The price to pay is an arbitrary choice for a assigning such pixels to one of the closest seeds. This phenomenon, which is true for the Voronoï tessellation of binary images directly translates to the watershed itself, as its construction is made by successive geodesic SKIZ. Such a partition is called watershed zone. If one consider a graph where the nodes are the pixels and the edges connect neighboring pixels, we obtain a partial graph connecting only pixels belonging to the same tile of the watershed partition. As there never exists an edge between two distinct tiles, this partial graph is a graph cut of the initial graph [14]. Such partitions could not easily be constructed through thinnings but their construction became easy with the apparition of general purpose computers with cheap memories, able to hold complete images.

3 Random access memories and waiting queue driven algorithms

Random access memories permit simulating the progression of a flood in a much more efficient way, as on hardwired devices, where the whole image has to be processed for each step progression of the flood. The first development uses a hierarchical queue controlling the propagation of labels for constructing a skeleton by zones of influence. This method permits to construct ad libitum skeletons by zone of influence or Voronoï tessellations and by replacing the thinnings in the first generation algorithms produced efficient watershed algorithms on general purpose computers. A hardwired implementation of this algorithm has been proposed in [49]. In order to be able to rapidly generate the successive thresholds of a grey tone image, L.Vincent and P.Soille had the idea to produce a histogram of the image in a first run and then to order the addresses of each pixel in bins with the right size for this particular grey tone. With these innovations, the algorithm of Lantuejoul could be implemented and gain new speed [57],[59].

The introduction of a hierarchical queue (HQ) for controlling the flood during the watershed construction presented a great advantage. It produces a correct flood not only from one grey tone to the next, but also within the flat-zones of the image. Furthermore, without modification, it is equally able to construct the watershed associated to all minima or to a set of markers [36].

4 The topographic distance and shortest path algorithms

This first period is dominated by algorithms and lack a precise definition of the watershed. Two independent papers introduced the topographic distance and defined the watershed as a SKIZ of the minima for this distance [47],[39]. The equidistant lines from the minima are the level lines of the topographic surface. This definition was thus compatible with the presentation of the watershed lines as dams to be erected for separating the floods from distinct minima during a flooding of a topographic surface [4]. Furthermore, it can be shown that the HQ algorithm directly derives from this definition [39].

As the geodesic lines of the topographic surface follow lines of steepest descent, another type of algorithms has been developed, where a graph is constructed linking each node with its lowest neighbors. This graph is then pruned in order to keep only one lower neighbor, creating a forest, where each tree spans a region of the partition. This idea has been used for parallelization of the watershed between various processors [9], [24] and for a hardwired implementation of the watershed [29].

The construction of the watershed may be then be obtained as a shortest path problem on a graph for which many algorithms exist [44],[2]. In order to obtain higher precision on digital grids, G.Borgefors introduced chamfer distances [10]. The same type of neighborhoods, based on particular weights for first and second neighbors on a grid can also be adapted for the construction of chamfer topographic distances [39]. Using a hierarchical queue for controlling the Dijkstra-Moore algorithm furthermore permits a correct flooding of the plateaus. Shortest path algorithms lend themselves also very well to the implementation on graphics processors or GBU [25]

These watershed algorithms may be subdivided in two classes : the first class constructs a watershed line separating connected particles ; the second produces a partition of the image, where each region represents a catchment basin.

The definition of the watershed line leads to an eikonal equation, expressed as a PDE and may be solved as such. This leads to a continuous watershed algorithm. [31],[32].

The so-called watersnakes, which introduce some degree of viscosity in order to regularize the watershed contours are also based on the topographic distance [48]. J. Roerdink published a remarkable review on the various methods for constructing the watershed [50].

5 Minimum spanning trees and forests, marker based segmentation

The segmentation paradigm based on watershed and markers has proved to be robust and efficient for solving many segmentation tasks. Its strength lies in the decoupling between a loose localization of the objects of interest, detected as markers and the precise construction of the contours. This advantage is particularly true in 3D, where the construction of the contours is complex, whereas

detecting the markers is often much simpler and may sometimes be done in 2D cuts of the 3D images.

Marker based segmentation is also ideal for interactive segmentation: a first set of markers obtained automatically or interactively introduced in the image produce a first segmentation. This segmentation may then be corrected by adding, modifying or suppressing markers. Adding a marker to an existing segmentation results in cutting a region of this segmentation in two parts. Suppressing a marker on the contrary results in merging two regions. As a matter of fact, marker based segmentation results in merging some of the catchment basins associated to the complete collection of minima of the image.

This leads to an approach where two scales are considered : for segmenting an image, the catchment basins of its gradient image are first constructed at the pixel level ; the final segmentation is then made at the level of regions. To this effect one constructs the region adjacency graph, where nodes represent the regions and edges link neighboring nodes. The edges are furthermore weighted by a weight expressing the dissimilarity between regions. As the boundaries of the regions follow the crest lines of a gradient image, one often expresses this dissimilarity by the altitude of the pass point between adjacent regions. This weighting is coherent with the flooding paradigm underlying the watershed : the propagation of a flooding in a topographic surface crosses the boundaries between catchment basins through their pass points. Flooding a topographic surface creates lakes. The lowest level of a lake containing two regional minima m_1 and m_2 of a topographic surface constitutes an ultrametric distance between these minima. If m_1, m_2 and m_3 are three minima, then the lowest lake covering all three minima is higher or equal than the lowest lake covering only two minima, constituting the ultrametric inequality $\max[d(m_1, m_2), d(m_2, m_3)] \geq d(m_1, m_3)$. The minimum spanning tree of the RAG is a tree spanning all nodes and whose total weight is minimal [12] [13] [11]. If the edge weights are all distinct, the minimum spanning tree is unique ; when several MSTs exist, they all have the same weight distribution.

MST constitute a sparse representation of a topographic surface as the number of edges equals to the number of nodes minus 1. Between any two nodes, there exists a unique path on the MST and the weight of the largest edge along this path is equal to the flooding ultrametric distance between these nodes (see the textbook [19]). Cutting all edges of the MST above some threshold produces a forest where each subtree spans a region of the domain. For higher thresholds, regions merge and coarser partitions produced. The series of nested partitions constitutes a hierarchy. If by cutting the edges above a given threshold produces n subtrees, they constitute a minimum spanning forest with n trees of the region adjacency graph. Marker based segmentation also produces minimum spanning forests with an additional constraint : each tree is rooted in a marker [38]. Marker based segmentation may also be formalized in terms of the SKIZ of the markers using a lexicographic distance [41].

6 From connected operators and floodings to hierarchies

The partition obtained by cutting the edges of the MST or of the RAG above some threshold is often not very useful as long it only relies on local dissimilarities between regions. Better focused segmentations may be obtained if one selectively floods some catchment basins before constructing the watershed line. Floodings have been introduced as reconstruction closings [16],[51] and subsequently generalized as levelings [40]. The watershed partition of an image produces a first segmentation ; flooding this image produces a coarser partition, where regions of the previous segmentation have merged. To each additional flooding of the preceding will correspond a coarser partition. The series of these partitions form a hierarchy. Such a hierarchy may be obtained in one run through the image, rather than repeating n increasing floodings and watershed basins detection.

M. Grimaud and L.Najman were the first to propose such a construction. At the time, M.Grimaud tried to detect the microcalcification in breast X rays ; they appear as small and contrasted bright dots. They appear on a fibrous substrate and coexist with noise particles. M.Grimaud wanted to rank all such events independently of the contrast of the image and measure additional features on the most contrasted ones. For this reason, he favoured a reconstruction closing sensitive to the contrast, where the marker is the function itself after the addition of a constant value λ. For increasing values of the constant λ, more and more basins will be filled and the subsequent watershed construction produce coarser segmentations. Each minimum can then be weighted by the parameter λ for which it is completely filled ; at the same time each contour can be weighted by the parameter λ for which it disappears for the first time. M.Grimaud proposed an algorithm for weighting all minima, calling the contrast measure dynamics [20]; on the other hand, L. Najman weighted the contours and called the measure saliency [45], [46].

Other criteria than the contrast may be used for governing the flooding of the basins. If one uses as floodings the area closings introduced by L.Vincent [58], one obtains hierarchies governed by size criteria; . More generally, one may flood the basins in such a way that the lakes which are created have in common, either the depth, or the area, or the volume of water [53],[54].

All these approaches have in common to use the same MST of the region adjacency graph. They take the MST with a given set of weights as input and output a new set of weights on the edges. Thanks to this common structure, efficient interactive segmentation toolboxes may be produced [61] . For instance minimum spanning forests with trees rooted in markers may be derived from the MST whatever its weight distribution.

In a hierarchy one goes from a fine to a coarse partition by merging adjacent regions. This operation is immediate if one deals with partitions : one assigns to all regions to be merged the same label. It is however more problematic if the contour is materialized between the regions and paradoxical situations may be met if one does not carefully chose the graph representing the images [15]. This is an additional reason why to prefer watershed zones without boundaries

between regions ; furthermore representing contours wastes space in the image and makes it impossible to segment adjacent small structures.

7 The waterfall hierarchy or graph cuts

S.Beucher introduced another type of hierarchy, expressing the nested structure of the catchment basins. In the RAG the edges are weighted but not the nodes. Marker based segmentations chooses a subset of the nodes and constructs a MSF where each tree is rooted in a node. S.Beucher considered the topography expressed by this graph and defined the regional minima as the maximal partial graphs whose internal edges have the same weight and whose adjacent edges have higher weights. Constructing a minimum spanning forest where each tree is rooted in one of these regional minima produces a coarser partition [33]. This partition itself may again by represented by a higher order RAG and MST on which the same procedure may be applied again. The corresponding hierarchy is called waterfall hierarchy [5].

Later J.Cousty also considered the problem of an edge weighted graph. He called the resulting MSF graph cut and proposed an efficient algorithm for constructing it [14].

7.1 Viscous and stochastic watershed

The watershed, being based on floodings is extremely sensitive to leaks in the topographic surface. For this reason, some works have attempted to regularize the watershed by introducing some viscosity. We already quoted the watersnakes [48]. Another approach consists in applying to the topographic surface an adaptive closing in order to produce a new surface on which the ordinary watershed flooding would progress in the same way as a viscous fluid would propagate in the initial topographic surface [55].

The classical use of the watershed is to find the contours associated to all minima or to a set of markers in a topographic surface. J.Angulo had the idea to weight the contours of the watershed by the probability they appear when random markers are used for segmenting the image. He called it the stochastic watershed [1].

8 Watershed : a name put in all sauces

This brief history of the development of the watershed concepts, construction algorithms and its use in the segmentation shows a contrasted and confusing picture. Distinct algorithms claim to produce watersheds, although they clearly produce distinct objects. The watershed may be topographic, viscous, stochastic, with or without apparent contours, defined on pictures where the nodes are weighted or on graphs where the edges are weighed. A number of issues are often not clearly addressed. The most annoying is the fact that one always speaks of

watershed lines, as if the watershed always is a line, at least in the continuous space. In fact, this is not at all the case, neither in images nor in the geology. There exist so called buttonholes which are large drainage zones whose outlet is a single point, at the same topographic distance of two minima. In this case, the complete buttonhole belongs to a thick watershed zone. If one decides to divide the buttonhole between these minima, is poses again the problem of the unicity of the watershed, as there are obviously many possibilities to perform this division ? There are objective reasons for the existence of multiple solutions. A drop of water falling inside a plateau has no clear indication in which direction to flow, if only local neighborhoods are considered. We also mentioned the non unicity of the MST of a RAG. What is the incidence of choosing one or another ?

Very often, definitions of watershed are given, without analyzing the unicity or multiplicity of solutions. Similarly does a particular algorithm give the same result if one changes the processing order. If several solutions may be produced by the same algorithm or be compatible with a given definition, are these solutions close one to another or in contrary extremely diverse ?

9 The history goes on...

This short history of the birth of the watershed for segmentation is necessarily uncomplete : google finds 31.000.000 entries for watershed.I hope that it is not biased although it puts a particular focus on the development over time of the concepts, algorithms and usage of the watershed at the CMM. This study is also interesting from an epistemological point of view : it shows that how a concept emerges, depending both on theoretical considerations (the study of the SKIZ by Lantuejoul for instance), the lessons of experience (nasty oversegmentation on real images, leading to the idea to introduce markers), technical revolutions (cheap memories permitting efficient random access to images, leading to the algorithms driven by hierarchical queues), etc.

Part II

Reminders on adjunctions and graphs

10 Reminder on adjunctions

The following section contains a reminder on adjunctions linking erosions and dilations by pairs and from which openings and closings are derived [52],[22],[21].

10.1 Adjunct pairs of dilation/erosion

Definition of adjunctions Let $\mathcal{T}$ a complete totally ordered lattice, where O is the smallest element and $\Omega$the largest element of $\mathcal{T}$ and let $\mathcal{D}$,$\mathcal{E}$ be arbitrary sets. Images will be functions:

- Fun($\mathcal{D}$,$\mathcal{T}$) : the image defined on the support $\mathcal{D}$ with value in $\mathcal{T}$
- Fun($\mathcal{E}$,$\mathcal{T}$) : the image defined on the support $\mathcal{E}$ with value in $\mathcal{T}$

We consider two operators α and β :
α : Fun($\mathcal{D}$,$\mathcal{T}$) $\rightarrow$ Fun($\mathcal{E}$,$\mathcal{T}$) and
β : Fun($\mathcal{E}$,$\mathcal{T}$) $\rightarrow$ Fun($\mathcal{D}$,$\mathcal{T}$).
Let f be a function of Fun($\mathcal{D}$,$\mathcal{T}$) and g be a function of Fun($\mathcal{E}$,$\mathcal{T}$):

Definition 1. *α and β form an adjunction if and only if :*
for any f in Fun($\mathcal{D}$,$\mathcal{T}$) *and g in* Fun($\mathcal{E}$,$\mathcal{T}$) *: $\alpha f < g \Leftrightarrow f < \beta g$*

Replacing $<$ by $=$ in the preceding relation would mean that α and β are inverse one from another. As we only have inequalities, they are only quasi-inverse, in a sense we will see later.

Erosion and dilation

Theorem 1. *If (α, β) form an adjunction, then*
α is a dilation (it commutes with the supremum of functions in Fun($\mathcal{D}$,$\mathcal{T}$))
and β is an erosion (it commutes with the infimum of functions in Fun($\mathcal{E}$,$\mathcal{T}$))

Proof. Let f be a function of Fun($\mathcal{D}$,$\mathcal{T}$) and $(g)_i$ functions of Fun($\mathcal{E}$,$\mathcal{T}$)
Then $f < \beta\bigwedge_i g_i \Leftrightarrow \alpha f < \bigwedge_i g_i \Leftrightarrow \forall i : \alpha f < g_i \Leftrightarrow \forall i : f < \beta g_i \Leftrightarrow f < \bigwedge_i \beta g_i$
Reading these equivalences from left to right and replacing f by $\beta\bigwedge_i g_i$ implies that $\beta\bigwedge_i g_i < \bigwedge_i \beta g_i$.
Reading these equivalences from right to left and replacing f by $\bigwedge_i \beta g_i$ implies that $\bigwedge_i \beta g_i < \beta\bigwedge_i g_i$
establishing that $\beta\bigwedge_i g_i = \bigwedge_i \beta g_i$.

Similarly we show that δ is a dilation, i.e. commutes with the union.

Remark 1. Calling O a constant function equal to O, we have for any $f : O < \beta f$ implying $\alpha O < f$. This relation is true for all f, indicating that $\alpha O = O$.
Similarly, we show that $\beta\Omega = \Omega$, where Ω is the constant function equal to Ω.

Lemma 1. *α and β are increasing operators.*

Proof. If $f < g$ then $\beta(f \wedge g) = \beta(f) = \beta(f) \wedge \beta(g) < \beta(g)$

From one operator to the other From the relation $\alpha f < g \Leftrightarrow f < \beta g$ one derives expressions of one operator in terms of the other one:

- If α is known, β may be expressed as $\beta g = \bigvee \{f \mid \alpha f < g\}$.
- Inversely if β is known, then $\alpha f = \bigwedge \{g \mid f < \beta g\}$

10.2 Opening and closing

Lemma 2. *The operator $\beta\alpha$ is a closing : increasing, extensive and idempotent. Similarly the operator $\alpha\beta$ is an opening : increasing, anti-extensive and idempotent.*

Proof. Openings and closings, obtained by composition of increasing operators, are increasing.
By adjunction we obtain $\alpha f < \alpha f \Rightarrow f < \beta\alpha f$ showing that the closing is extensive and $\alpha\beta f < f \Longleftarrow \beta f < \beta f$ showing that the opening is anti-extensive.
In particular, applying an opening to αf yields $\alpha f > \alpha\beta\alpha f$.
On the other hand, α being increasing, and the closing extensive :
$f < \beta\alpha f \Rightarrow \alpha f < \alpha\beta\alpha f$ showing that $\alpha f = \alpha\beta\alpha f$
Applying β on both sides yields $\beta\alpha f = \beta\alpha\beta\alpha f$

The family of invariants of an opening or a closing We call $\mathrm{Inv}(\gamma)$ and $\mathrm{Inv}(\varphi)$ the family of invariants of γ and φ.

Lemma 3. *The family of invariants of an opening is closed by union. And the family of closings is closed by intersection.*

Proof. Let us prove it for openings ; the result for closings being obtained by duality. Suppose that g_1 and g_2 are invariant by the opening γ : $\gamma(g_1) = g_1$ and $\gamma(g_2) = g_2$.
Then $\gamma(g_1 \vee g_2) \leq g_1 \vee g_2 = \gamma(g_1) \vee \gamma(g_2)$ by antiextensivity
And $\gamma(g_1 \vee g_2) \geq \gamma(g_1) \vee \gamma(g_2)$ as γ is increasing.

In the proof given above that $\alpha\beta$ is an opening and $\beta\alpha$ a closing, we have established :

- $\alpha f = \alpha\beta\alpha f = (\alpha\beta)\alpha f$ which implies that $\alpha f \in \mathrm{Inv}(\gamma)$
- $\beta f = \beta\alpha\beta f = (\beta\alpha)\beta f$ which implies that $\beta f \in \mathrm{Inv}(\varphi)$

The opening as pseudo-inverse operator of the erosion We already made the remark that replacing $<$ by $=$ in the adjunction relation $\alpha f < g \Leftrightarrow f < \beta g$ would mean that α and β are inverse one from another. As a matter of fact, they are not and we only have $\alpha\beta g \leq g$ and $\beta\alpha g \geq g$. However, we have the following lemma:

Lemma 4. *The opening is the pseudo inverse of the erosion : $\alpha\beta g$ is the smallest function having the same erosion as g. Similarly the closing is the pseudo inverse of the dilation : $\beta\alpha g$ is the largest function having the same dilation as g*

Proof. Suppose that $f \leq \alpha\beta g$ verifies $\beta f = \beta g$ which implies $\alpha\beta f = \alpha\beta g$. Since $f \leq \alpha\beta g$, we have $\alpha\beta f \leq f \leq \alpha\beta g$ showing that $f = \alpha\beta g$. Hence $\alpha\beta g$ indeed is the smallest function having the same erosion then g. For this reason, we give the name pseudo-inverse : the erosion has no inverse, but the family of functions with an identical erosion has a smallest element, which is $\alpha\beta g$.

Remark 2. If $g \in \mathrm{Inv}(\gamma)$, we have $\gamma g = \alpha\beta g = g$: on $\mathrm{Inv}(\gamma)$, $\alpha = \beta^{-1}$. If $g \in \mathrm{Inv}(\varphi)$, we have $\varphi g = \beta\alpha g = g$: on $\mathrm{Inv}(\varphi)$, $\beta = \alpha^{-1}$

11 Weighted graphs

11.1 Reminders on graphs

Graphs

Basic definitions A *non oriented graph* $G = [N, E]$ is a collection N of vertices or nodes and of edges E, an edge $u \in E$ being a pair of vertices (see [2],[19]).

A chain of length n is a sequence of n edges $L = \{u_1, u_2, \ldots, u_n\}$, such that each edge u_i of the sequence $(2 \leq i \leq n-1)$ shares one extremity with the edge u_{i-1} $(u_{i-1} \neq u_i)$, and the other extremity with u_{i+1} $(u_{i+1} \neq u_i)$.

A *path* between two nodes x and y is a sequence of nodes $(n_1 = x, n_2, ..., n_k = y)$ such that two successive nodes n_i and n_{i+1} are linked by an edge.

A cycle is a chain or a path whose extremities coincide.

A cocycle is the set of all edges with one extremity in a subset Y and the other in the complementary set $\overline{Y}$.

The subgraph spanning a set $A \subset N$ is the graph $G_A = [A, E_A]$, where E_A are the edges linking two nodes of A.

The partial graph associated to the edges $E' \subset E$ is $G' = [N, E']$.

For *contracting* an edge (i, j) in a graph G, one suppresses this edge u and its two extremities are merged into a unique node k. All edges incident to i or to j become edges incident to the new node k.

Connectivity *A connected graph* is a graph where each pair of nodes is connected by a path.

A tree is a connected graph without cycle.

A *spanning tree* is a tree containing all nodes.

A forest is a collection of trees.

Labelling a graph means extracting the maximal connected subgraphs and assigning to each of them a different label.

Contraction : If H is a connected subgraph of G, we call $\kappa : (G, H) \rightarrow G'$ the operator which contracts all edges of H in G.

Weighted graphs In a graph $G = [N, E]$, edges and nodes may be weighted : e_{ij} is the weight of the edge (i, j) and n_i the weight of the node i. The weights take their value in the completely ordered lattice $\mathcal{T}$

Weights may be attributed to edges only, to nodes only or to both. Hence the following weight distributions are possible, and we write:

$G \in (-, \diamond)$ if G has no weights at all

$G \in (-, n)$ if G has its weights only on the nodes

$G \in (e, \diamond)$ if G has its weights only on the edges

$G \in (e, n)$ if G has its weights on both edges and nodes

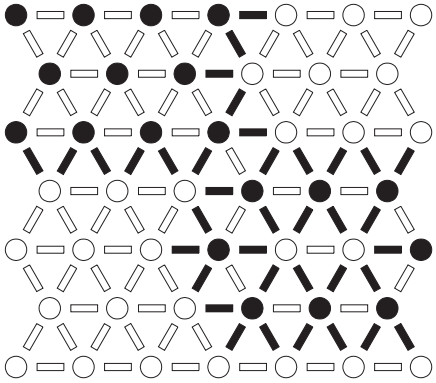

Fig. 1. Graph obtained by linking neighboring pixels of an image by an edge

11.2 Which types of graphs ?

In "pixel graphs" the nodes are the pixels and the edges connect neighboring pixels. The weights of the pixels are their value and the weights of edges may be for instance a gradient value computed between their extremities. Fig.1 represents the graph associated to a binary image ; edges linking two extremities with the same value have a weight equal to 1, the others a weight equal to 0. The connected components of this graph represent the connected particles and connected holes of the binary image.

Neighborhood graphs are frequently met in the context of image segmentation. The nodes represent the tiles of a partition, edges connecting adjacent tiles. The edge weights generally represent a dissimilariry between the regions represented by the extremities of the edge. Morphological segmentation frequently uses the watershed transform which associates to a topographical surface its catchment basins. Two catchment basins are neighbors if there exists a pass point for passing from one to the other. The altitude of this pass point constitutes a frequently used edge weight (see fig.2)

Gabriel and Delaunay graphs permit to define neighborhood relations between the elements of a population of points are of sets. The Gabriel graph is defined for points. Two nodes p and q are linked by an edge if there exists no other node in the disk of diameter pq as illustrated in fig. 3. Node weights will express features of the nodes, edge weights relationships between adjacent nodes. The equivalent graph for sets is called perceptual graph [34]

11.3 Minimum spanning trees and forests in an edge weighted graph

A *minimum spanning tree* is a spanning tree for which the sum of the edges is minimal.
A *spanning forest* is *a collection of trees spanning all nodes.*

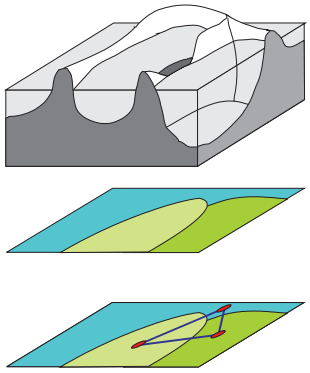

Fig. 2. Flooding a topographic surface in order to detect its catchment basins. Representation of the neighborhood graph.

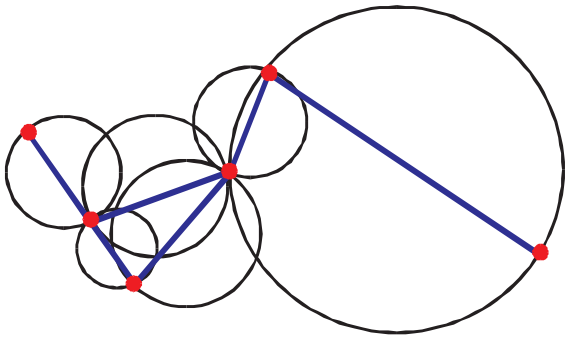

Fig. 3. Two points p and q are connected if there is no third point falling in a disk of diameter pq.

In a *minimum spanning forest,* the sum of the edges is minimal.
The *minimum minimorum spanning forest (MSF)* has only nodes. Wih an additional constraint, one gets particular forests:

- a MSF with a fixed number of trees, useful in hierarchical segmentation.
- a MSF where each tree is rooted in predefined nodes, useful in marker based segmentation. [38]

11.4 Flat zones and regional minima on edge weighted graphs

Edge weighted graphs

Definition 2. *A subgraph G' of an edge weighted graph G is a flat zone, if any two nodes of G' are connected by a chain of uniform altitude.*

Definition 3. *A subgraph G' of a graph G is a regional minimum if G' is a flat zone and all edges in its cocycle have a higher altitude.*

We define an operator $\mu_e : G \rightarrow \mu_e G$ extracting all regional minima of the graph G

Node weighted graphs

Definition 4. *A subgraph G' of a node weighted graph G is a flat zone, if any two nodes of G' are connected by a path where all nodes have the same altitude.*

Definition 5. *A subgraph G' of a graph G is a regional minimum if G' is a flat zone and all neighboring nodes have a higher altitude.*

We define an operator $\mu_n : G \rightarrow \mu_n G$: extracting all regional minima of the graph G.

11.5 Other operators on graphs

Contracting or expanding graphs Contraction: For *contracting* an edge (i, j) in a graph G, one suppresses this edge u and its two extremities are merged into a unique node k. All edges incident to i or to j become edges incident to the new node k and keep their weights.

If H is a subgraph of G, the operator $\kappa : (G, H) \rightarrow G'$ contracts all edges of each connected component of H in G. Graph contraction will be used below for highlighting the nested structure or hierarchy of the waterfall partitions.

Expansion: The operator $\multimap : (-, n) \rightarrow \multimap G$ creates for each isolated regional minimum i a dummy node with the same weight, linked by an edge with i. It permits to create graphs where each regional minimum node is linked with a regional minimum edge.

Extracting partial graphs We also need various operator for pruning graphs by suppressing particular edges. The following operators suppress a subset of the edges in a graph:

$\chi : G \in (-, \diamond) \rightarrow \chi G$ keeps for each node only one adjacent edge.

$\downarrow : G \in (e, \diamond) \rightarrow \downarrow G$ keeps for each node only its lowest adjacent edges.

$\Downarrow : G \in (-, n) \rightarrow \Downarrow G$ keeps only the edges linking a node with its lowest adjacent nodes.

These operators may be concatenated:

$\chi \downarrow G$ keeps for each node only one lowest adjacent edge.

$\chi \Downarrow G$ keeps for each node only one edge linking it with one of its lowest adjacent nodes.

12 Distances on a graph

12.1 Case of edge weighed graphs

The weights are assigned to the edges, and represent their altitudes.

Method for constructing a distance on an edge weighted graph Distances on an edge weighted graph have chains as support and are defined with the following steps:

- definition of the weight of a chain, as a measure derived from the edge weights of the chain elements (example : sum, maximum, etc.).
- An order relation is defined between chains. Two chains being compared by their weights: the chain with the smallest weight is called the shortest.

The distance $d(x, y)$ between two nodes x and y of a graph is ∞ if there is no chain linking these two nodes and equal to the weight of the shortest chain if such a chain exists.

Triangular inequalities appear quite naturally with the concatenation of chains.

Given three nodes (x, y, z) the concatenation of the shortest chain π_{xy} between x and y and the shortest chain π_{yz} between y and z produces a chain π_{xz} between x and z, whose weight is larger or equal to the weight of the shortest chain between x and z. To each distance corresponds a particular triangular inequality : $d(x, z) = weight(\pi_{xz}) \leq weight(\ \pi_{xy} \triangleright \pi_{yz})$ where $\pi_{xy} \triangleright \pi_{yz}$ represents the concatenation of both chains.

Distance on an edge weighted graph based a the length of the shortest chain Length of a chain: The length of a chain between two nodes x and y is defined as the sum of the weights of its edges.

Distance: The distance $d(x, y)$ between two nodes x and y is the minimal length of all chains between x and y. If there is no chain between them, the distance is equal to ∞. (see fig. 4)

Triangular inequality : For $(x, y, z) : d(x, z) \leq d(x, y) + d(y, z)$

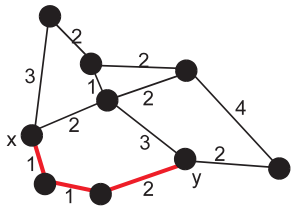

Fig. 4. The shortest chain (sum of the weights of the edges) between x and y is a red line and has a length of 4. It is also the lowest chain (maximum of the weights of the edges) with a maximal weight of 2.

Distance on a graph based on the maximal edge weight along the chain
Altitude of a chain: The altitude of a chain is equal to the highest weight of the edges along the chain.

Flooding distance between two nodes: A flood having its source at one extremity of the chain reaches the other extremity as soon as its level reaches the level of the highest edge along the chain. For this reason we call the associated distance flooding distance. The flooding distance fldist(x, y) between nodes x and y is equal to the minimal altitude of all chains between x and y. If there is no chain between them, the flooding distance is equal to ∞. (see fig. 4)

Triangular inequality : For $(x, y, z) : d(x, z) \leq d(x, y) \vee d(y, z)$: it expresses that the lowest lake containing both x and y is lower or equal than the lowest lake containing x, y and z.

The flooding distance is an ultrametric eccart, as it verifies in addition :

* reflexivity : $d(x, x) = 0$

* symmetry: $d(x, y) = d(y, x)$

It is however not a distance as it does not verify: $d(x, y) \Rightarrow x = y$

The lexicographic distance or the cumulative effort for passing the highest edges Toughness of a chain: We call toughness of a chain the decreasing list of altitudes of the highest edges when travelling along this chain. If one travels from the origin of the chain towards the extremity, the first element of the toughness is the altitude of the highest edge met along the path. When this edge is crossed, the highest edge on the remaining path to the extremity constitutes the second element of the toughness. The same process is repeated until one arrives at destination. More formally, the toughness $\Lambda(A)$ of a chain $A = e_{12}e_{23}...e_{n-1n}$ is constructed as follows. Following the chain from the origin towards its end, one records the highest valuation of the chain, let it be λ_1, then again the highest valuation λ_2 on the remaining part of the chain and so on until the end is reached. One gets like that a series of non increasing values: $\lambda_1 \geq \lambda_2 \geq ....\geq \lambda_n$. The toughness is not symmetrical.

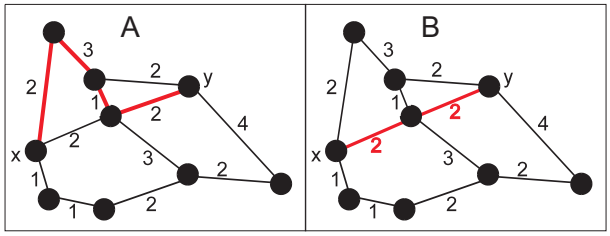

Fig. 5. Lexicographic distances. The toughness of the red chain in fig.A in the direction from x to y is $[3, 2]$: after crossing the highest edge at altitude 3, the next highest edge has altitude 2. The easiest chain from x to y it indicated in red in fig.B, its toughness is $[2, 2]$.

Remark 3. The toughness of a never increasing path (NAP), is simply equal to the series of weights of its edges. Later we introduce shortest path algorithms whose geodesics are NAPs.

We now introduce a total preorder relation permitting to compare the toughness of two lexicographic chains. We first consider chains for which the toughness list has the same length.

The lexicographic preorder relation of length k compares the lists $\pi = (\lambda_1, \lambda_2, ..., \lambda_k)$ and $(\mu_1, \mu_2, ..., \mu_k)$

* $\pi \prec^k \chi$ if $\lambda_1 < \mu_1$ or there exists $t < k$ such that $\begin{array}{l} \forall l < t : \lambda_l = \mu_l \\ \lambda_t < \mu_t \end{array}$

* $\pi \preceq^k \chi$ if $\pi \prec^k \chi$ or if $\forall l \leq k : \lambda_l = \mu_l$.

This preorder relation is total, as it permits to compare all lists with the same length. The same definition is valid for $k = \infty$. We write $\prec$ for $\prec^\infty$ and $\preceq$ for $\preceq^\infty$.

If we have to compare two lists with different lengths, we prolongate them by duplicating infinitely the last element of the list. The list $(\lambda_1, \lambda_2, ..., \lambda_n)$ becomes $(\lambda_1, \lambda_2, ..., \lambda_n, \lambda_n, \lambda_n ...)$. Like the preorder is total,i.e. is able to compare any pair of lists.

Lexicographic distance: The lexicographic distance between two nodes p and q on an edge weighted graph is equal to the toughness of the chain going from p to q with the smallest toughness for the relation $\preceq$. Note that this distance is not a value but a non increasing list of values (see fig. 5).

Remark 4. Lexicographic distances have been used in the context of marker based segmentation on edge valued graph: each region is the set of nodes which are closer for the lexicographic distance to a node than to any other node [41].

The lexicographic distance of depth k between two nodes x and y is written $\text{lexdist}_k(x, y)$ and obtained by retaining only the k first edges in $\text{lexdist}(x, y)$. The distance $\text{lexdist}_1(x, y)$ is the same as the ultrametric flooding distance as it is the highest edge on the lowest path between x and y.

12.2 Case of node weighed graphs

The weights are assigned to the nodes, and represent their altitudes.

Method for constructing distances on a node weighted graph. Distances on a node weighted graph have paths as support and are defined in two steps:

- Definition of the "length" of a path, as a measure derived from the node weights of the path elements (example : sum, maximum, etc.)
- Comparison of two paths by their length. The path with the smallest length is called the shortest.

The "distance" $d(x, y)$ between two nodes x and y of a graph is ∞ if there is no path linking these two nodes and equal to the length of the shortest path if such a path exists.

Given three nodes (x, y, z) the concatenation of the shortest path π_{xy} between x and y and the shortest path π_{yz} between y and z is not a path, as the node y is counted twice. For establishing the triangular inequalities, one has to suppress this node counted twice.

Distance on a node weighted graph based on the maximal node weight along the path Altitude of a path: The altitude of a path is equal to the highest weight of the nodes along the path.

Flooding distance between two nodes: The flooding distance $\text{fldist}(x, y)$ between nodes x and y is equal to the minimal altitude of all paths between x and y.

Distance on a node weighted graph based on the cost for travelling along the cheapest path Each node may be considered as a town where a toll equal to its weight has to be paid.

Cost of a path: The cost of a path is equal to the sum of the tolls to be paid in all towns encountered along the path (including or not one or both ends).

Cost between two nodes: The cost for reaching node y from node x is equal to the minimal cost of all paths between x and y. We write $\text{tolldist}(x, y)$. If there is no path between them, the cost is equal to ∞. (see fig. 6)

Remark 5. The toll distance on a node weighted graph, is the same as the distance on an edge weighted graphs. The edges are oriented, and the weight of each edge from i to j is equal to the weight of the node j. Like that, whatever the edge which has been crossed for arriving at j, the same toll has to be paid, equal to the weight of j.

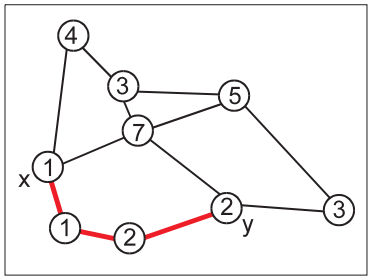

Fig. 6. The cheapest chain between x and y is in red and the total toll to pay is $1+1+2+2=6$.

Part III

The flooding adjunction and the flooding graph

13 Two adjunctions between edges and nodes on weighted graphs

We consider graphs without isolated nodes.

13.1 Definition of two dual adjunctions

Definition 6. *We define two operators between edges and nodes :*
- an erosion $[\varepsilon_{en} n]_{ij} = n_i \wedge n_j$ *and its adjunct dilation* $[\delta_{ne} e]_i = \bigvee_{(k \, neighbors \, of \, i)} e_{ik}$
- a dilation $[\delta_{en} n]_{ij} = n_i \vee n_j$ *and its adjunct erosion* $[\varepsilon_{ne} e]_i = \bigwedge_{(k \, neighbors \, of \, i)} e_{ik}$

Lemma 5. *The operators we defined are pairwise adjunct or dual operators:*
- ε_{ne} *and* δ_{en} *are adjunct operators*
- ε_{en} *and* δ_{ne} *are adjunct operators*
- ε_{ne} *and* δ_{ne} *are dual operators*
- ε_{en} *and* δ_{en} *are dual operators*

Proof. 1) Let us prove that δ_{en} and ε_{ne} are adjunct operators
If $G = [e, n]$ and $\overline{G} = [\overline{e}, \overline{n}]$ are two graphs with the same nodes and edges, but with different valuations on the edges and the nodes then
$\delta_{en} n \leq \overline{e} \Leftrightarrow \forall i, j : n_i \vee n_j \leq \overline{e}_{ij} \Leftrightarrow \forall i, j : n_i \leq \overline{e}_{ij} \Leftrightarrow \forall i, j : n_i \leq \bigwedge_{(j \, \text{neighbors of } i)} \overline{e}_{ij} =$
$[\varepsilon_{ne} \overline{e}]_i \Leftrightarrow n \leq \varepsilon_{ne} \overline{e}$
which establishes that δ_{en} and ε_{en} are adjunct operators.
2) Let us prove that ε_{en} and δ_{en} are dual operators
$[\varepsilon_{en}(-n)]_{ij} = -n_i \wedge -n_j = -(n_i \vee n_j) = -[\delta_{en} n]_{ij}$
Hence $\delta_{en} n = -[\varepsilon_{en}(-n)]$

13.2 The flooding adjunction

The dilation $[\delta_{en} n]_{ij} = n_i \vee n_j$ and its adjunct erosion $[\varepsilon_{ne} e]_i = \bigwedge_{(k \, \text{neighbors of } i)} e_{ik}$ have a particular meaning in terms of flooding. If n_i and n_j represent the altitudes of the nodes i and j, the lowest flood covering i and j has the altitude $[\delta_{en} n]_{ij} = n_i \vee n_j$ (see fig. 9). If i represents a catchment basin, e_{ik} the altitude of the pass points with the neighboring basin k, then the highest level of flooding without overflow through an adjacent edge is $[\varepsilon_{ne} e]_i = \bigwedge_{(k \, \text{neighbors of } i)} e_{ik}$.

This erosion has a particular meaning for neighborhood graphs, where the nodes represent the catchment basins of a topographic surface and the edges connect neighboring basins. The waterfall flooding of a topographic surface consists in filling each catchment basin up to its lowest pass point. It is the highest flooding possible in this basin without overflow (the flooding of a one dimensional topographic surface is illustrated in fig. 10). The waterfall flooding consists thus in assigning to each node the weight of its lowest adjacent edges, i.e.

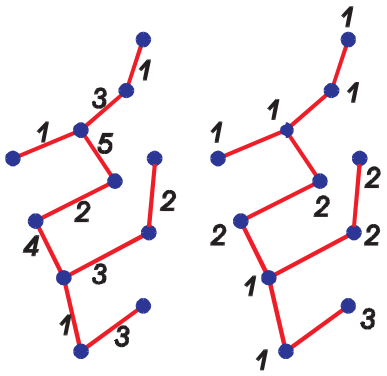

Fig. 7. On the left an edge weighted graph. On the right the result of the erosion between edges and nodes.

it is an erosion between the edges and the nodes on the neighborhood graph.: $[\varepsilon_{ne} e]_i = \bigwedge_{(k \text{ neighbors of } i)} e_{ik}$ (see fig. 6 and fig. 8).

13.3 Erosion between edges and edges / between nodes and nodes

By concatenation of operators between edges and nodes we obtain :

- an adjunction between nodes and nodes : $(\varepsilon_{ne}\varepsilon_{en}, \delta_{ne}\delta_{en}) = (\varepsilon_n, \delta_n)$
- an adjunction between edges and edges : $(\varepsilon_{en}\varepsilon_{ne}, \delta_{en}\delta_{ne}) = (\varepsilon_e, \delta_e)$

The erosions associated to these adjunctions will play an important role later, for propagating the values of nodes and edges along the paths of steepest descent.

14 Opening and closing

As ε_{ne} and δ_{en} are adjunct operators, the operator $\varphi_n = \varepsilon_{ne}\delta_{en}$ is a closing on n and $\gamma_e = \delta_{en}\varepsilon_{ne}$ is an opening on e.

In the sequel, the flooding adjunction will play a key role, in particular the associated opening γ_e and closing φ_n.

Similarly the operator $\varphi_e = \varepsilon_{en}\delta_{ne}$ is a closing on e and $\gamma_n = \delta_{ne}\varepsilon_{en}$ is an opening on n.

14.1 The flooding opening γ_e

Construction and illustration Fig.10 presents a one dimensional topographic surface and its neighborhood graph below. The nodes represent the catchment basins ; the nodes representing adjacent basins are linked by edges weighted with the altitude of the pass points separating the basins. The maximal flooding of

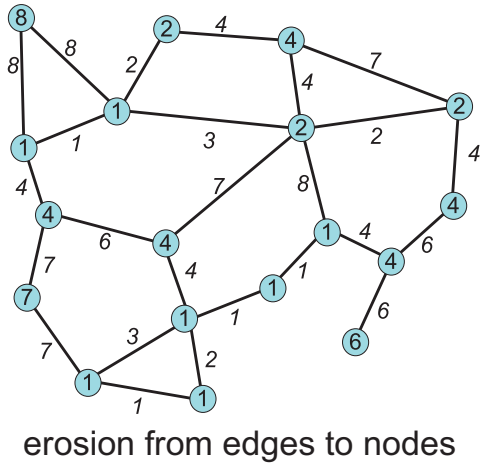

Fig. 8. The edge weights are given and the node weights are the result of the erosion between edges and nodes

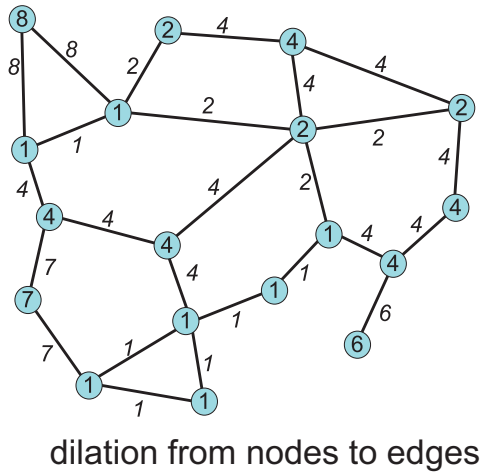

Fig. 9. The node weights are given and the edge weights are the result of the dilation between edges and nodes

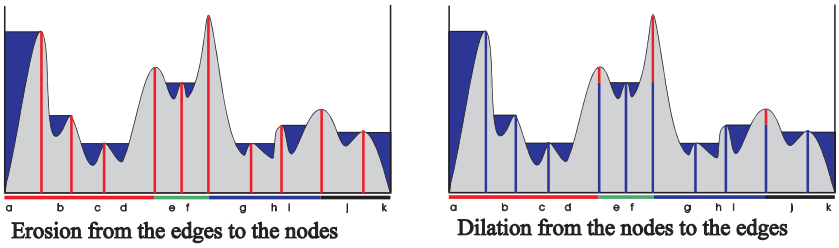

Fig. 10. On the left: waterfall flooding by an erosion between edges and nodes ; on the rights subsequent dilation between nodes and edges yielding an opening.

a basin before an overflow occurs reaches its lowest pass point ; such a flooding where each basin is filled up to its lowest pass point is called waterfall flooding. In terms of graphs, it corresponds to an erosion between edges and nodes. The subsequent dilation restores most of the edges to their original altitude but not all. The altitude to which the edges are restored is indicated in blue ; the edges which are not restored to their initial altitude have 2 colors, the result of the opening in blue, the initial altitude in red. It is easy to check, and we will prove it below, that the corresponding passpoints are those which are not the lowest pass points of a neighboring catchment basin.

Fig.11 shows an edge weighted graph and its erosion between edges and nodes. A final dilation between nodes and edges produces the opening γ_e. The edges which are restored to their original weight are in blue. Those which are lowered are in red, they are all edges which are not the lowest edge for one of their extremities.

The invariants of the opening γ_e Let us analyze under which conditions an edge is invariant by the opening γ_e. Two possibilities exist for an edge (i, j) with a weight λ :

- the edge (i, j) has lower neighboring edges at each extremity. Hence $\varepsilon_{ne}(i) < \lambda$ and $\varepsilon_{ne}(j) < \lambda$; hence $\delta_{en}\varepsilon_{ne}(i, j) = \varepsilon_{en}(i) \vee \varepsilon_{en}(j) < \lambda$.
- the edge (i, j) is the lowest edge of the extremity i. Then $\varepsilon_{ne}(i) = \lambda$ and $\varepsilon_{ne}(j) \leq \lambda$; hence $\delta_{en}\varepsilon_{ne}(i, j) = \varepsilon_{en}(i) \vee \varepsilon_{en}(j) = \lambda$.

This analysis shows that the edges invariant by the opening γ_e are the edges which are the lowest edge of one of their extremities. All edges with lower adjacent edges at both of their extremities have their weight lowered by the opening γ_e. They play no role in the erosion between edges and nodes. On the contrary, if the edge (i, j) is the lowest edge of the node i, then this node i takes the weight e_{ij} after the erosion ε_{ne}.

The relation $\varepsilon_{ne} = \varepsilon_{ne}(\delta_{en}\varepsilon_{ne})$ shows that all edges which are not invariant by the opening $\gamma_e = \delta_{en}\varepsilon_{ne}$ play no role in the erosion. As a matter of fact, if the opening lowers the valuation of an edge (i, j) and if the subsequent erosion

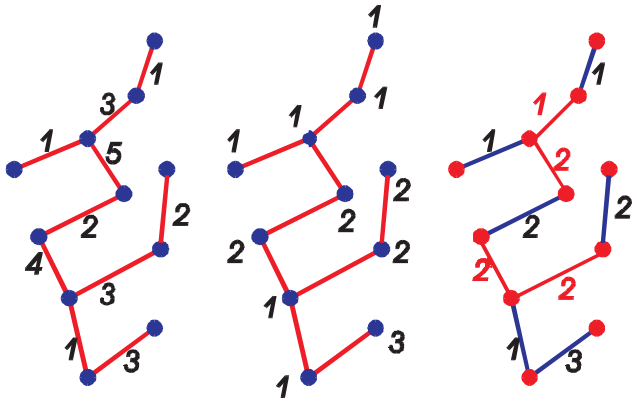

Fig. 11. From left to right: 1) an edge weighted graph, in the centre, 2) the result of the erosion ε_{ne}, 3) the subsequent dilation produces an opening. The edges in red are those whose weight has been reduced by the opening as they were not the lowest edge of one of their extremities in the initial distribution of edge weights.

ε_{ne} is not modified, it means that initial weight of this edge plays no role in the erosion ε_{ne}.

The drainage graph, partial graph invariant by the opening γ_e Suppressing in an arbitrary graph g all edges which are not invariant by the opening γ_e produces a graph g', invariant for the opening γ_e and called **drainage graph**. The pruning operator keeping for each node only its lowest adjacent edges and suppressing all others is written $\downarrow : (e, \diamond) \rightarrow \downarrow G$ and transforms each graph into its drainage graph. The drainage graph has the following properties.

- $\downarrow G$ spans all the nodes (each node has at least one lowest neighboring edge ; such edges are invariant by γ_e).
- And $\varepsilon_{ne}(G) = \varepsilon_{ne}(\downarrow G)$ (ε_{ne} assigns to each node the weight of its lowest adjacent edge, which is the same in G as in $\downarrow G$).
- the operator $\downarrow$ may disconnect a connected graph and create several connected components. However, it cannot create isolated nodes, as it leaves at least one adjacent edge for each node.

Fig. 12 illustrates the construction of the drainage graph. On the left an edge weighted graph ; the weights of the nodes are deduced from the weights of the edges by the erosion ε_{ne}. The central image illustrates the opening of the edge weights, obtained by applying the dilation δ_{en} to the node weights. On the right only the edges invariant by the opening are kept, the others are suppressed, resulting in the drainage graph, which now has several connected components.

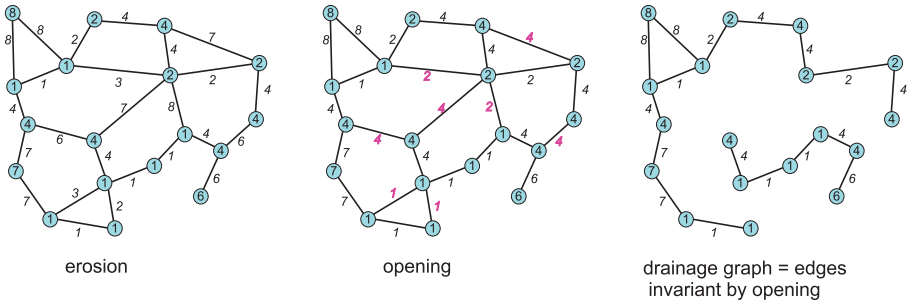

Fig. 12. Erosion from edges to nodes on the left, subsequent dilation producing the opening γe in the centre and drainage graph on the right where only the edges invariant by the opening are kept.

Identity of the erosion from edges to nodes on a graph and its drainage graph The relation $\varepsilon_{ne} = \varepsilon_{ne}(\delta_{en}\varepsilon_{ne})$ shows that all edges which are not invariant by the opening $\gamma_e = \delta_{en}\varepsilon_{ne}$ play no role in the erosion. Suppressing all these edges does not change the resulting erosion between edges and nodes. In other words we get the same distribution of weights on the nodes if we apply the erosion ε_{ne} to a graph or to its drainage graph. Fig. 13_A shows a graph, fig.13_B the distribution of weights on the nodes after the erosion ε_{ne}.Fig. 13_C shows the associated drainage graph and fig. 13_D the the distribution of weights on the nodes after the erosion ε_{ne} of the drainage graph. Both erosions produce the same distribution of weights on the nodes.

Fig. 14 shows another illustration of the identity of results: $\varepsilon_{ne}(G) = \varepsilon_{ne}(\downarrow G)$

The invariants of the opening γ_e, or how to get drainage graphs The family of openings is closed for the supremum. Hence if a graph G has two edge weight distributions $(e_1, \diamond)$ and $(e_2, \diamond)$, which are invariant for γ_e, then the distribution $(e_1 \vee e_2, \diamond)$ also is invariant for γ_e.

An arbitrary graph may be transformed into a drainage graph:

- For edge weighted graphs, by applying the opening γ_e to the edge weight distribution $(e, \diamond) : (\gamma_e e, \diamond) \in \mathrm{Inv}(\gamma_e)$ or by by applying the operator $\downarrow$ suppressing all edges not invariant by the opening γ_e.
- For node weighted graphs $(-, n)$ the dilation δ_{en} produces a drainage graph as $\gamma_e\delta_{en}n = \delta_{en}\varepsilon_{ne}\delta_{en}n = \delta_{en}n$, showing that $(\delta_{en}n, n) \in \mathrm{Inv}(\gamma_e)$

If a connected and edge weighted graph $G = (E, N)$ is invariant by γ_e then :

- any subgraph G' of G spanning a subset A of the nodes N, but keeping the same edge weights belongs to $\mathrm{Inv}(\gamma_e)$. An edge of G' has both extremities in A ; being also an edge of G, it is the lowest edge of one of its extremities.

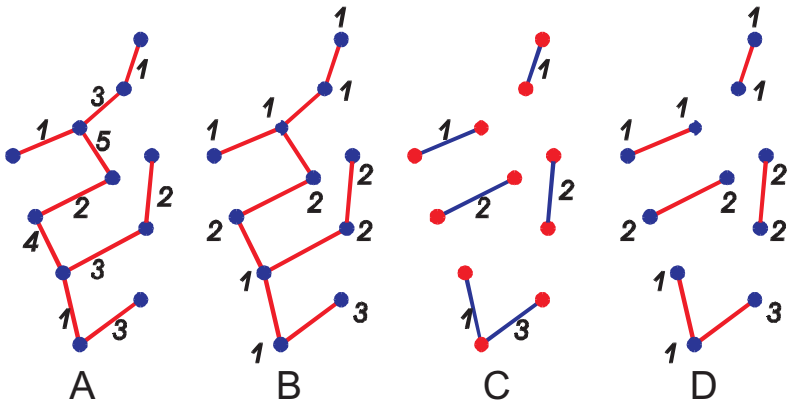

Fig. 13. A: Initial edge weighted graph G
B: Erosion ε_{ne} from the edges to the nodes on G
C: The graph $\downarrow G$
D: Erosion ε_{ne} from the edges to the nodes on $\downarrow G$: $\varepsilon_{ne}(G) = \varepsilon_{ne}(\downarrow G)$

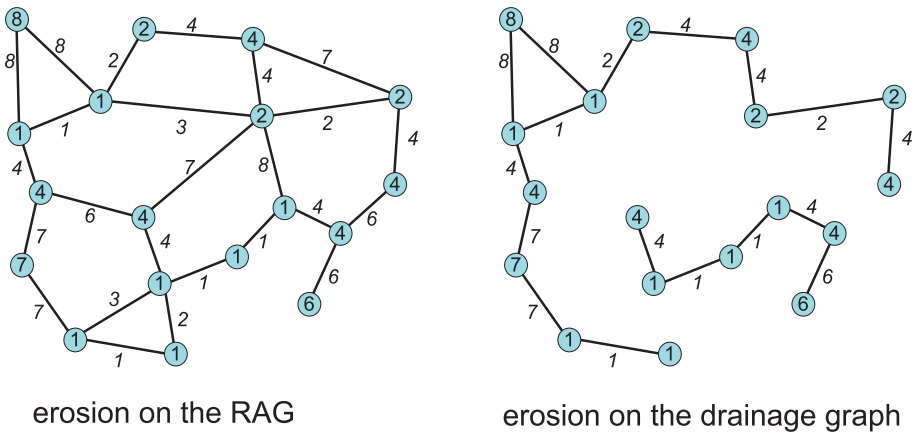

Fig. 14. Another illustration of $\varepsilon_{ne}(G) = \varepsilon_{ne}(\downarrow G)$

Fig. 15. A graph where the right most node with weight 1 is an isolated node regional minimum without belonging to an edge regional minimum. On the contrary the left most edge with weight 2 is an edge regional minimum spanning two node regional minima.

– any partial graph, containing only a subset of the edges of E, with the same weights also belongs to $\mathrm{Inv}(\gamma_e)$. In particular $\downarrow : G = (e, \diamond) \rightarrow \downarrow G$ containing for each node only its lowest adjacent edges and $\chi \downarrow G$ keeping only one lowest adjacent edge for each node.

Inverse of ε_{ne} on the invariants of the opening γ_e On $\mathrm{Inv}(\gamma_e) : \delta_{en}\varepsilon_{ne} = \mathit{Identity}$ showing that on $\mathrm{Inv}(\gamma_e) := \delta_{en} = \varepsilon_{ne}^{-1}$

The regional minima of the opening γ_e

Theorem 2. *If an edge weighted graph $G = (e, \diamond)$ is invariant by the opening γ_e, and m is the edge weighted subgraph of its regional minima, then $\varepsilon_{ne}m \in (-, \varepsilon_{ne}e)$ is the node weighted subgraph of the regional minima of the node weighted graph $\varepsilon_{ne}G \in (-, \varepsilon_{ne}e)$*

Proof: A regional minimum m_k of the graph $G = (e, \diamond) \in \mathrm{Inv}(\gamma_e)$ is a plateau of edges with altitude λ, with all external edges having a weight higher than λ. If a node i belongs to this regional minimum, its adjacent edges have a weight $\geq \lambda$ but it has at least one neighboring edge with weight λ : hence $\varepsilon_{ne}e(i) = \lambda$. Consider now an edge (s, t) outside the regional minimum, with the node s inside and the node t outside the minimum. Then $e_{st} > \lambda$. As $G = (e, \diamond) \in \mathrm{Inv}(\gamma_e)$, the edge (s, t) is then one of the lowest edges of the nodes t : thus $\varepsilon_{ne}(t) = e_{st} > \lambda$. This shows that the nodes spanned by the regional minimum m_k form a regional minimum of the graph $\varepsilon_{ne}G = (-, \varepsilon_{ne}e)$

Remark 6. If $G = (e, \diamond) \in \mathrm{Inv}(\gamma_e)$, then $e = \delta_{en}\varepsilon_{ne}e$. The node and edge weighted graph $(e, \varepsilon_{ne}e)$ verifies then $n = \varepsilon_{ne}e$ and $e = \delta_{en}\varepsilon_{ne}e = \delta_{en}n$. Such graphs are called flooding graphs and are studied below in detail.

Remark 7. Inversely, given a node weighted graph $(-, n)$, the edge and node weighted graph $(\delta_{en}n, n)$ is invariant by the opening γ_e and the previous theorem applies : the nodes spanned by an edge regional minimum form a node regional minimum of the graph. Each such node regional minimum contains at least two nodes. The graph may contain additional regional minima which are isolated nodes and whose adjacent edges do not belong to a regional minimum. Fig.15 gives an example of such a situation.

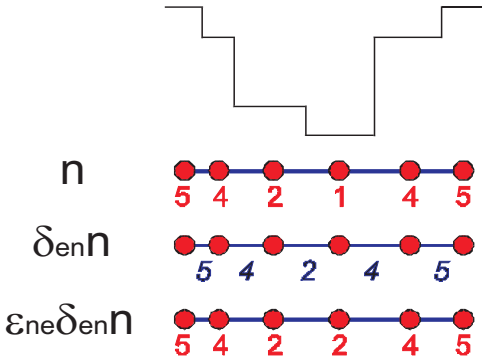

Fig. 16. Illustration of the closing φ_n, obtained by an erosion ε_{ne} following a dilation δ_{en}. All nodes remain unchanged, except the isolated regional minima, which take the value of their lowest neighboring node.

14.2 The closing ϕ_n

We consider here node weighted graphs without isolated nodes. The closing φ_n is obtained by a dilation δ_{en} of the node weights followed by an erosion ε_{ne} as illustrated by fig.16. One remarks in fig.16 that the node weights remain the same, except the isolated regional minima, which take the weight of their lowest neighboring node.

Two possibilities exist for a node i with a weight λ. The dilation δ_{en} assigns to each edge with extremity i a weight $\geq \lambda$.

- the node i is an isolated regional minimum. Then δ_{en} assigns to all edges adjacent to i a weight higher than λ. The subsequent erosion ε_{ne} assigns to i the smallest of these weights, which is higher than λ.
- the node i has a neighbor j with a weight $\mu \leq \lambda$. Then δ_{en} assigns to the edge (i, j) the weight λ. Hence all edges with extremity i have a weight $\geq \lambda$ and at least one has a weight equal to λ. The subsequent erosion ε_{ne} assigns to i the weight λ.

Hence the closing φ_n replaces each isolated node constituting a regional minimum by its lowest neighboring node and leaves all other nodes unchanged.

Applying the erosion ε_{ne} to an arbitrary edge weighted graph also produces a graph invariant by φ_n as $\varphi_n \varepsilon_{ne} e = \varepsilon_{ne}\delta_{en}\varepsilon_{ne} e = \varepsilon_{ne} e$; the resulting graph has no isolated regional node minima.

Property of the minima: If $G = (-, n) \in \mathrm{Inv}(\varphi_n)$ and $m = (-, n)$ is the subgraph of its regional minima, then $\delta_{en} m = (\delta_{en} n, \diamond)$ is the subgraph of the regional minima of the graph $\delta_{en} G = (\delta_{en} n, \diamond)$

Inverse of δ_{en} on the invariants of the closing φ_n On $\mathrm{Inv}(\varphi_n) : \varepsilon_{ne}\delta_{en} = \mathit{Identity}$ showing that on $\mathrm{Inv}(\varphi_n) : \varepsilon_{ne} = \delta_{en}^{-1}$

Creating invariants graphs for the closing φ_n For an arbitrary node weight distribution $(-, n)$ the simplest way to obtain an invariant for the closing φ_n is to apply this closing to the node weights. Obviously$(-, \varphi_n n) \in \text{Inv}(\varphi_n)$. Unfortunately, this modifies the weight of the isolated regional minima and extends them by fusion with their lower neighboring plateaus. If we want to keep the node weights unchanged, we have to apply the "expansion operator" presented earlier, which for an arbitrary node weight distribution $g = (-, n)$ creates an "expanded graph" $\multimap g$ by creating for each isolated regional minimum i a dummy node with the same weight linked by an edge with i. This new graph, having no isolated regional minima anymore is now invariant by the closing $\varphi_n : \multimap g \in \text{Inv}(\varphi_n)$.

Alternatively, if we add an edge linking i with itself, we create a loop edge, which gets a weight λ by the dilation δ_{en}. And the subsequent erosion ε_{ne} assigns to i its initial weight λ. In what follows we equip each isolated regional minimum by such a loop edge ; we transform lile that the node weighted graph $G = (-, n)$ into the graph $\circlearrowleft G = (-, \circlearrowleft n)$ invariant by φ_n.

The family of invariants of a closing is closed for the infimum. If $(-, n_1)$ and $(-, n_1)$ are two node weight distributions which are invariant for φ_n, then $(-, n_1 \wedge n_2)$ also is invariant for φ_n.

Applying to an edge weighted graph $(e, \diamond)$ the erosion ε_{ne} creates a distribution of node weights invariant by φ_n. As a matter of fact: $(e, \varepsilon_{ne} e) \in \text{Inv}(\varphi_n)$ as $\varphi_n \varepsilon_{ne} e = \varepsilon_{ne} \delta_{en} \varepsilon_{ne} e = \varepsilon_{ne} e$. As a matter of fact: $(e, \varepsilon_{ne} e) \in \text{Inv}(\varphi_n)$ as $\varphi_n \varepsilon_{ne} e = \varepsilon_{ne} \delta_{en} \varepsilon_{ne} e = \varepsilon_{ne} e$.

As a consequence, applying to a node weighted graph $(-, n)$ the erosion $\varepsilon_n = \varepsilon_{ne} \varepsilon_{en}$ creates a distribution of node weights invariant by φ_n.

Obviously, any partial or subgraph of a node weighted graph $G \in \text{Inv}(\varphi_n)$, without solated regional minima also belongs to $\text{Inv}(\varphi_n)$.

In particular, a node weighted graph G invariant by φ_n remains invariant by φ_n after the pruning $\Downarrow G$ which keeps only the edges linking a node with its lowest neighbors. Each node i belonging to a regional minimum of g is linked with at least another node j in this minimum through an edge. Such an edge is not pruned by $\Downarrow g$ as it links i with one of its lowest neighboring nodes.

The regional minima of the closing φ_n

Theorem 3. *If $G = (-, n) \in \text{Inv}(\varphi_n)$ and $m = (-, n)$ is the subgraph of its regional minima, then $\delta_{en} m = (\delta_{en} n, \diamond)$ is the subgraph of the regional minima of the graph $\delta_{en} G = (\delta_{en} n, \diamond)$*

Proof: A regional minimum m_i of a graph $G = (-, n) \in \text{Inv}(\varphi_n)$ is a plateau of pixels with altitude λ, containing at least two nodes (there are no isolated regional minima in $\text{Inv}(\varphi_n)$). All internal edges of the plateau get the valuation λ by $\delta_{en} n$. If an edge (i, j) has the extremity i in the minimum and the extremity j outside, then $\delta_{en} n(i, j) > \lambda$. Hence, for the graph $(\delta_{en} n, \diamond)$, the edges spanning the nodes of m_i form a regional minimum.

15 The flooding graphs

The literature on watersheds is itself divided by a divide line: on one side the watersheds on node weighted graphs, on the other side the watersheds on edge weighted graphs. Flooding graphs introduced below show that this division makes no sense as flooding graphs offer a perfect coupling between edge and node weights: one set of weights may be deduced from the other, and the catchment basins associated to either one or the other are the same.

15.1 Definition and basic properties

Definition 7. *An edge and node weighted graph $G = [N, E]$ is a flooding graph iff its weight distribution (n, e) verifies the relations:*
- $\delta_{en} n = e$
- $\varepsilon_{ne} e = n$

Corollary 1. *For a flooding graph, the weight distribution $(n, e) verifies$:*
- $n \in \mathrm{Inv}(\varphi_n)$
- $e \in \mathrm{Inv}(\gamma_e)$

Proof. $e = \delta_{en} n = \delta_{en} \varepsilon_{ne} e = \gamma_e e$ and $n = \varepsilon_{ne} e = \varepsilon_{ne} \delta_{en} n = \varphi_n n$

Properties of the flooding graph As G is invariant by γ_e, all its edges are the lowest edge of one of their extremities.

As G is invariant by φ_n, it has no isolated regional minimum.

Lemma 6. *In a flooding graph, a node has no adjacent edges with a lower weight but each node has at least one adjacent edge with the same weight.*

Proof. In a flooding graph, as $\varepsilon_{ne} e = n$, all edges adjacent to a node i have weights which are higher or equal than this node and at least one of them, say (i, j) has the same weight : $e_{ij} = n_i$.

15.2 Constructing flooding graphs.

Flooding graphs offer a perfect coupling between the edge and the node weights. As a matter of fact, constructing the watershed for node weighted graphs or constructing it for edge weighted is strictly equivalent, since, as we will show below, any node weighted graph without edge weights, or any edge weighted graph without node weights may be completed to become a flooding graph, on which the watershed will be ultimately constructed.

Transforming an arbitrary edge weighted graph into a flooding graph

Lemma 7. *If* $G = (e, \diamond) \in \mathrm{Inv}(\gamma_e)$, *then* $(e, \varepsilon_{ne} e)$ *is a flooding graph*

Proof. a) $e = \gamma_e e = \delta_{en} \varepsilon_{ne} e = \delta_{en} n$) ; b) $n = \varepsilon_{ne} e$ by construction.

Lemma 8. *If* $G = (e, \diamond)$ *is an arbitrary edge weighted graph, then* $(\downarrow e, \varepsilon_{ne} \downarrow e)$ *is a flooding graph*

Proof. If $G = (e, \diamond)$ is an arbitrary edge weighted graph, then $\downarrow (e, \diamond) \in \mathrm{Inv}(\gamma_e)$, as the edges lowered by γ_e, have been suppressed, the other edges keep their weights. Furthermore we have proved above that $\varepsilon_{ne} e = \varepsilon_{ne} \downarrow e$. Applying the previous lemma shows that $(\downarrow e, \varepsilon_{ne} \downarrow e)$ is a flooding graph.

Transforming an arbitrary node weighted graph into flooding graph

Lemma 9. *If* $G = (-, n) \in \mathrm{Inv}(\varphi_n)$, *then* $(\delta_{en} n, n)$ *is a flooding graph*

Proof. a) $e = \delta_{en} n$) by construction ; b) $n = \varphi_n n = \varepsilon_{ne} \delta_{en} n = \varepsilon_{ne} e$.

Lemma 10. *If* $G = (-, n)$ *is an arbitrary node weighted graph, then* $(\delta_{en} \multimap n, \multimap n)$ *is a flooding graph*

Proof. If $G = (e, \diamond)$ is an arbitrary node weighted graph, then $(-, \multimap n) \in \mathrm{Inv}(\varphi_n)$, as each isolated regional minimum has been duplicated by the operator $\multimap n$. Applying the preceding lemma shows that $(\delta_{en} \multimap n, \multimap n)$ is a flooding graph.

Partial graph of a flooding graph Suppressing edges in a flooding graph, but leaving at least one lower neighboring edge for each node (like that, no isolated regional minima are created) produces a partial graph which also is a flooding graph, keeping an identical distribution of weights on the nodes and on the remaining edges.

15.3 Regional minima of flooding graphs

We proved earlier these theorems:

- If $G \in \mathrm{Inv}(\gamma_e)$ is an edge weighted graph, and m_i is an edge regional minimum of G, then the nodes spanned by m_i form a node regional minimum of the node weighted graph $\varepsilon_{ne} G = (-, \varepsilon_{ne} e)$
- If $G \in \mathrm{Inv}(\varphi_n)$ is a node weighted graph, and m_i is a node regional minimum of G, then the edges spanning m_i form an edge regional minimum of the edge weighted graph $\delta_{en} G = (\delta_{en} n, \diamond)$

As in a flooding graph $n = \varepsilon_{ne} e$ and $e = \delta_{en} n$ we derive:

Theorem 4. *If G is a flooding graph with the weight distribution (e, n), then the node weighted graph $(-, n)$ and the edge weighted graph $(e, \diamond)$ have the same regional minima subgraph.*

Fig.17 presents the same flooding graph, on the left with its edge weights and on the right with its node weights : they have exactly the same regional minima.

The minima of a flooding graph being identical for the edge weigths and for the node weights may be assigned the same labels. The labels may be hold by the edges of the minima or by their extremities, or by both, as illustrated by fig.18.

15.4 Perfect equivalence between node and edge weights on flooding graphs

We now understand why the watershed of a flooding graph may be constructed on the basis of the node weights or on the basis of the edge weights : they produce the same result :

- the node weights and the edge weights may be derived from each other
- the minima are the same
- each node has no lower adjacent edges but at least one with the same weight. A node and the adjacent edge with the same weight is called flooding pair. A series of non increasing flooding pairs is both a non increasing path of nodes and a non increasing chain of edges.

We define the catchment basin of a regional minimum as the set of nodes which are linked by a non increasing path for node weighted graphs and a non increasing chain for edge weighted graphs with this minimum. As the minima are the same and the non increasing paths or chains also are equivalent, this shows that indeed the watershed constructed on the node weights or on the edge weights of a flooding graph are the same.

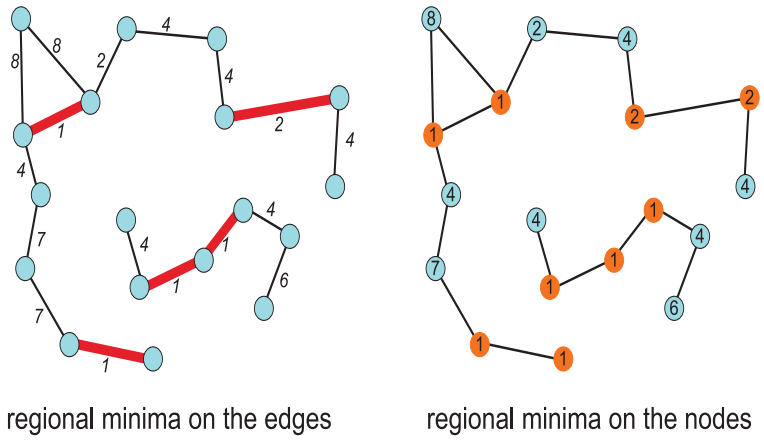

Fig. 17. Whether one consider the egde weights or the node weights produces the same regional minima.

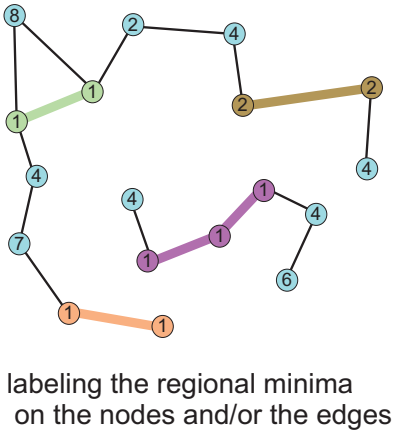

Fig. 18. As the minima are identical on the nodes or the edges of a flooding graph, it is possible to assign the same labels to nodes or to edges.

Part IV

The catchment basins

16 Steep, steeper and steepest descent paths and catchment basins

16.1 A lexicographic order relation between downwards paths

Given a node or weighted graph we first derive from it a flooding graph $G = [E, N]$, where the lowest edges of each node have the same weight. We associate to G an oriented graph $\overrightarrow{G}$ by replacing the edge (p, q) by an arrow $\overrightarrow{pq}$ if $n_p \geq n_q$ and *by* two arrows $\overrightarrow{pq}$ and $\overrightarrow{qp}$ if $n_p = n_q$. The loop edge linking an isolated regional minimum node m with itself also is replaced by an arrow $\overrightarrow{mm}$. The graph $\overrightarrow{G}$ verifies the property (P): $\forall p \in N$, there exists at least an oriented path of $\overrightarrow{G}$ (with a positive or null length) linking p with a regional minimum.

Definition 8. *We define the catchment basin of a regional minimum as the set of nodes linked by an oriented path with this minimum.*

Obviously, each node belongs to at least one catchment basins. Catchment basins may overlap and form a watershed zone when two paths having the same node as origin reach two distinct regional minima. We aim at pruning the graph $\overrightarrow{G}$, without any arbitrary choices, and get a partial graph $\overrightarrow{G'}$ for which the property (P) still holds but the watershed zones are smaller

We now define a family of preorder relations (order relation without anti-symmetry) between the paths of $\overrightarrow{G}$. The relations become simpler if we consider paths of infinite length, obtained by prolongating the paths inside the regional minima. If the regional minimum has more than one node, the prolongation moves endlessly from node to node and back inside the regional minimum ; if the regional minimum is isolated, then the path endlessly circles along the loop arrow linking this regional minimum node with itself.

The lexicographic preorder relation of length k compares the infinite paths $\pi = (p_1, p_2, ...p_k, ...)$ and $\chi = (q_1, q_2, ...q_k, ...)$:

* $\pi \prec^k \chi$ if $n_{p_1} < n_{q1}$ or there exists $t < k$ such that $\begin{array}{l} \forall l < t : n_{p_l} = n_{q_l} \\ n_{p_t} < n_{q_t} \end{array}$

* $\pi \preceq^k \chi$ if $\pi \prec^k \chi$ or if $\forall l \leq k : n_{p_l} = n_{q_l}$.

This preorder relation is total, as it permits to compare all paths ; for this reason, among all paths linking a node p with a regional minimum, there exists always at least one which is the smallest for $\preceq^k$.

For $k = \infty$, we consider the infinite paths, ending with constant values inside the regional minima. If π andχ are two paths of infinite length verifying $\pi \preceq \chi$, then the paths π_l andχ_l obtained by skipping the l first nodes also verify $\pi_l \preceq \chi_l$. If π is the smallest path linking its origin with a regional minimum, then π_l is the smallest path leading from p_{l+1} to the same regional minimum. Such a path is the steepest everywhere: if it enters or starts from a flat zone, it quits it as fast as possible.

For $k = \infty$, a node is linked by two minimal paths with two distinct minima, only if these two paths have exactly the same weights, which seldom happens in

natural images. Hence, if the regional minima have distinct weights, the catchment basins form a partition.

Nested catchment basins

Consider two lexicographic order relations $\prec^k$ and $\prec^l$ with $l > k$, then for $_1$ and $\pi_2 : \pi_1 \prec^k \pi_2 \Rightarrow \pi_1 \prec^l \pi_2$ or equivalently $\pi_1 \succeq^l \pi_2 \Rightarrow \pi_1 \succeq^k \pi_2$: the steepest path for the lexicographic order l also is steepest for the order k. As a consequence, a catchment basin for $\succeq^l$ is included in the catchment basin for $\succeq^k$. For increasing values of k, the catchment basins become larger, are nested, and the watershed zones are reduced or vanish.

16.2 Pruning the flooding graph to get steeper paths.

Pruning the flooding graph using non local operators We associate to each order relation $\preceq^k$ of length k a pruning operator $\downarrow^k$. The pruning $\downarrow^k$ suppresses each edge which is not the first edge of a path minimal for $\preceq^k$ among all paths with the same origin .After pruning, each node outside the regional minima is the origin of one or several $k-$flooding tracks or $k-$flooding paths with maximal steepness. We say that the graph $\downarrow^k \overrightarrow{G}$ has a k-steepness or is k-steep. As for $l > k$, $\pi_1 \succeq^l \pi_2 \Rightarrow \pi_1 \succeq^k \pi_2$, we have $\downarrow^l \overrightarrow{G} \subset \downarrow^k \overrightarrow{G}$. Furthermore $\downarrow^k \downarrow^l \overrightarrow{G} = \downarrow^l \downarrow^k \overrightarrow{G} = \downarrow^{k \vee l} \overrightarrow{G}$.

Particular k-steep graphs

Applied to an arbitrary graph, the pruning $\downarrow^1 = \downarrow$ suppresses the edges which are not the lowest edge of one of their extremities. In a flooding graph, each edge it the lowest edge of one of its extremities and $\downarrow^1$ is inoperant. The pruning $\downarrow^2$ keeps for each node i the adjacent edges linking i with one of its lowest neighboring nodes. The pruning $\downarrow^\infty$ only keeps the first edge of the steepest paths.

Lemma 11. *Any oriented path in $\downarrow^k \overrightarrow{G}$ of length k is of maximal steepness for $\preceq^k$.*

For this reason, a node p belongs to a k-catchment basin or k-CB associated to a node m in a regional minimum, if there exists an oriented path in $\downarrow^k \overrightarrow{G}$ from p to m.

Pruning the flooding graph using local operators The operator $\downarrow^k$ is not local. However it may be implemented with local operators. One operator is the pruning operator $\Downarrow \overrightarrow{G} = \downarrow^2 \overrightarrow{G}$ which keeps the edges linking a node with its lowest neighbors. The second is the erosion, assigning to each node the weight of its lowest neighboring node $\varepsilon = \varepsilon_{ne}\varepsilon_{en}$. Both operators transform a flooding graph into a flooding graph.

We illustrate the method on an example.

We first apply the operator $\downarrow^2$ to the flooding graph and get a graph where each node is only linked with its lowest neighbors like in the following graph: $0 \leftarrow 3 \leftarrow 4 \leftarrow 5 \rightarrow 4 \rightarrow 2 \rightarrow 1$ where each node is only linked with its lowest neighbors. The node 5 is the origin of two paths, π with weights $5 \rightarrow 4 \rightarrow 2 \rightarrow 1$ and π' with weights $0 \leftarrow 3 \leftarrow 4 \leftarrow 5$, verifying $\pi \prec^3 \pi'$.

The erosion ε assigns to each node the weight of its lowest neighbors: $\varepsilon\overrightarrow{G} = * \leftarrow 0 \leftarrow 3 \leftarrow 4 \rightarrow 2 \rightarrow 1 \rightarrow *$. Thus π with weights $4 \rightarrow 2 \rightarrow 1$ and π' with weights $0 \leftarrow 3 \leftarrow 4$ verify $\pi \prec^2 \pi'$. Applying the pruning operator $\downarrow^2$ suppresses the arc linking the node with weight 4 with the node with weight 3 in π' producing the graph $\downarrow^2 \varepsilon\overrightarrow{G} = * \leftarrow 0 \leftarrow 3 \quad 4 \rightarrow 2 \rightarrow 1 \rightarrow *$. This example shows that the operator $\downarrow^2 \varepsilon \downarrow^2 \overrightarrow{G}$ suppresses the same edges as the pruning operator $\downarrow^3 \overrightarrow{G}$.

Defining $\zeta = \downarrow^2 \varepsilon$ and $\zeta^{(k)} = \zeta\zeta^{(k-1)}$, one proves that $\zeta^{(k-2)} \downarrow^2$ and $\downarrow^k$ operate the same prunings on the edges of a graph $\overrightarrow{G}$.

Pruning the graph and labelling the catchment basins At the same time as we recursively apply the operator ζ we are able to construct and label the catchment basins. Consider a flooding graph G. The algorithm is the following:

* construct an oriented graph $\overrightarrow{G}$ by linking by an arc each node of G outside the regional minima with each of its lowest neighbors. Detect, label the regional minima and replace their inside arcs by edges. Figure 19_B represents the oriented flooding graph of fig.19_A.

* Repeat until stability : a) erode the graph $\overrightarrow{G}$ and cut all arcs linking a node to another which is not one of its lowest neighbors ; b) if (i, j) is an oriented arc from i to j, and if j holds a label whereas i has no label, then i is assigned the label of j, the arc between i and j is replaced by an edge and all arcs with origin i are suppressed. If i points to several nodes with distinct labels, one of them is chosen and the same procedure applied. This will be the first time where we introduce an arbitrary choice. No other operator presented so far does such an arbitrary choice.

Applied on fig.19_B the erosion and pruning produces fig.19_C ; the label propagation produces fig.19_D . The next iteration produces fig. 19_E and 19_F. After the last erosion, pruning and final label propagation, all nodes are labeled and the steepest drainage graph obtained. The final partition is superimposed on the initial graph in fig.19_H.

16.3 The scissor operator and the watershed partitions

Starting with an arbitrary node or edge weighted graph G we are able to derive an oriented graph $\overrightarrow{G}$ on which only oriented paths with a given steepness remain. None of the operators for constructing $\overrightarrow{G}$ makes an arbitrary choice between equivalent edges or equivalent paths. Increasing the steepness reduces the number of paths and reduces the watershed zones where neighboring catchment basins overlap. However, some catchment basins may nevertheless overlap. If we target

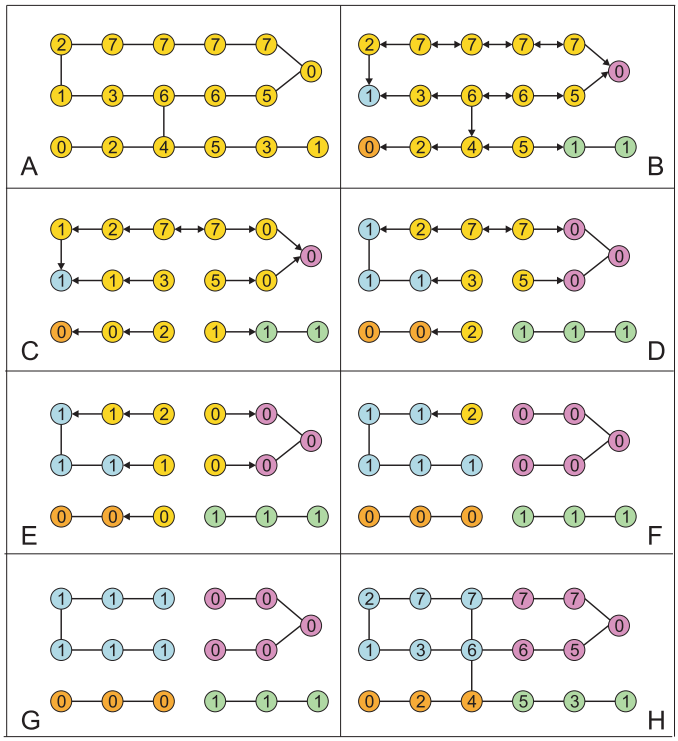

Fig. 19. Construction of the catchment basins of a node weighted graph

a partition of catchment basins, we have to introduce some more or less arbitrary choices between equivalent solutions. In order to help finding the best choice, some additional criteria may be used, as for instance favoring large regions or contrasted ones.

Playing with the levels of the minima

If the minima have all distinct altitudes, then the paths linking a node with two distinct minima cannot be equal under an infinite lexicographic order. Pruning $\overrightarrow{G}$ with $\zeta^{(\infty)}$ and modifying slightly the altitude of the minima in order to render the all different produces a partition of catchment basins. Playing with the altitude of the minima is of course arbitrary.

The scissor pruning χ

We now introduce an additional, arbitrary pruning operator χ. If a node is the origin of several arcs, the scissor operator χ leaves only one. If there exists an arc from i to j and another arc from j to i, then one of them at most is kept. After such a pruning there exists a unique path of length positive or null linking each node to a regional minimum. The catchment basins are then necessarily disjoint. The situation $i \longmapsto j$ or $j \longmapsto i$ only arrives if $n_i = n_j$, that is in the flat zones of the graph. For increasing values of k the pruning $\zeta^{(\infty)}$ reduces the

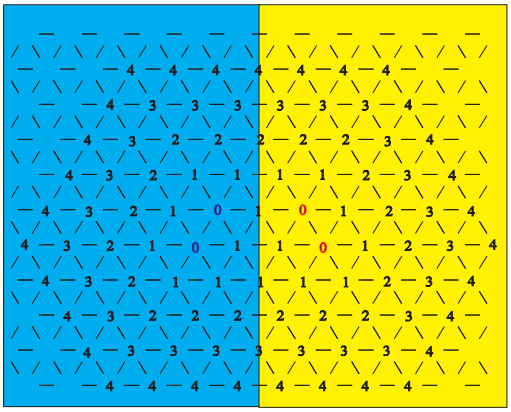

Fig. 20. Hexagonal distance function to two binary sets.

number of such situations. For $k = \infty$, they do not arrive. For $k = \infty$, the unique situation where a choice is to made is $p \leftarrow q \rightarrow s$ where $q \geq p, s$, and p and s are the first nodes of two absolutely identical non ascending paths.

The scissor pruning χ on plateaus.

One creates a geodesic distance on the plateaus : for each node in the plateau is computed the distance to the nearest lower neighbor of the plateau. For each node i inside a plateau, the operator χ keeps one edge linking i with a neighboring node j veriying $\delta_i > \delta_j$, where δ_i and δ_j are the geodesic distance functions of i and j to the lower borders of the plateaus.

The scissor pruning χ through flooding

The regional minima are labeled. Repeat until all nodes have a label: if i has a label, j has no label, $j \longmapsto i$, then assign to j the label of i and keep the arc $j \longmapsto i$ as only arc with j as origin.

Discussion: The choice of each new node to flood is completely arbitrary and many choices possible, some leading to absurd partitions as illustrated in the next paragraph.

16.4 Illustration

A flooding graph associated to a distance function We chose as topographic surface the distance function expressed on the nodes of a hexagonal grid to two binary connected sets, encoded with the value 0 (see fig.20). The edge weights being obtained by the dilation δ_{en}, . the resulting graph is invariant by γ_e. Like that the lowest neighboring edges of a node connects it with its neighbors with equal or lower weights. In fig. 21, the edges with the same weight have the same color (cyan = 4, magenta = 3, green = 2).

As the minima have 2 nodes each, the pixel graph is invariant by φ_n. The dilation δ_{en} assigns weights to the edges and creates a flooding graph.

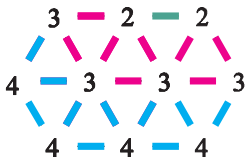

Fig. 21. Edge weights produces by the dilation δ_{en} of the hexagonal distance function on a small portion of the image.

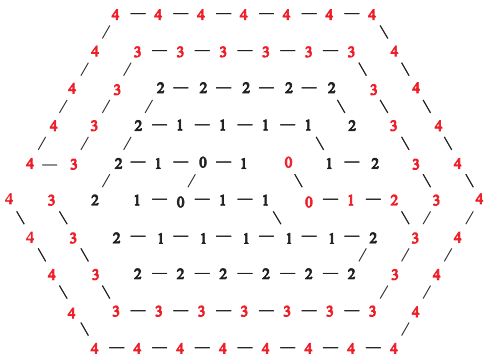

Fig. 22. An unexpected minimum spanning forest produced by the scissor operator χ, in one only considers 1-steepness.

The minimum spanning forest by keeping one lowest neighboring edge for each node The scissor χ leaves one lowest neighboring edge for each node. There exists a huge number of choices for χ. Fig.22 shows a particular scissor χ producing an unexpected partition in two catchment basins (the catchment basins should be the half plane separated by the mediatrix of both binary sets, as illustrated in the previous slide).

Fig.23 shows two flooding tracks leading to the regional minima.

Looking one node further The operator $\downarrow^2 G$ leaves only the edges linking a node to its lowest neighbors. Fig.24 shows the remaining edges. The green zone represents a restricted catchment basin. The yellow zone is an extended catchment basin containing the watershed zone.

The pruning $\chi \Downarrow = \chi \downarrow^2$ leaves for each node one edge towards a lowest neighboring node. Fig.25 shows a possible minimum spanning forest and the associated partition.

"Much ado about nothing" or the silent efficiency of hierarchical queues Hierarchical queues are ideal structures for controlling the flooding [36]. This data structure is a series of prioritized FIFOs. It permits to enqueue each

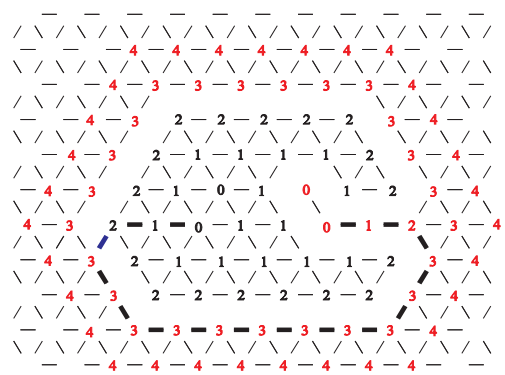

Fig. 23. two flooding tracks leading to the regional minima.

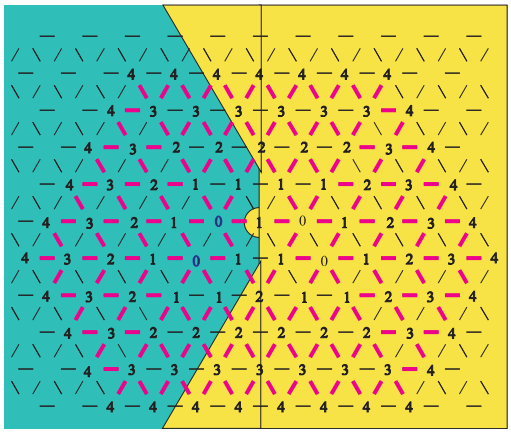

Fig. 24. 2-steep flooding graph, obtained by keeping only the edges linking each node with its lowest neighbors.

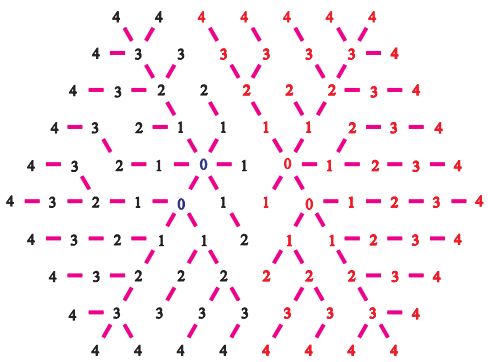

Fig. 25. A minimum spanning forest produced by the operator $\chi \Downarrow = \chi \downarrow^2$

node p in the FIFO with the priority ν_p. Dequeueing on the contrary consists in extracting the node which entered first into the FIFO with the highest priority. As the nodes close to the lower border of the plateau enter the queue before the inside nodes, the nodes inside the plateaus are indeed flooded in an order corresponding to increasing distance to the lower border of the plateau. Controlled by a hierarchical queue (HQ) the algorithm becomes

Label the nodes of the minima and introduce them in the HQ, each with a priority equal to its weight.

As long as the HQ is not empty, extract the node j with the highest priority from the HQ:

For each unlabeled neighboring (on the flooding graph) node i of j :

* $label(i) = label(j)$
* put i in the queue with priority n_i

This algorithm does more as simply correctly processing the inside of the plateaus. When a node is extracted from the hierarchical queue, it gives its label to all its neighbors without label. Like that, each node gets its label from one of its lowest neighbors ; in other terms, the flooding follows the arcs of a graph which has been pruned by $\downarrow^2$: it floods the surface in an order proportional to the topographic distance to the minima [47],[39].

Analyzing hierarchical queues with more care shows that in fact they even do more: they flood the nodes in the lexicographic order of infinite depth, i.e. they follow the arcs of $\downarrow^\infty$. Consider the fig.26 illustrating 2 steps the flooding of a node weighted graph. Initially the three regional minima are labeled and put into the hierarchical queue, with a lexicographic depth equal to their altitude. Fig. 26A presents an intermediate step of the flooding. All nodes which have been introduced in th HQ are present ; the nodes in red have not flooded their neighborhood yet, whereas the nodes in black have. On top of each group of nodes with the same lexicographic distance to a minimum we indicate this distance. In fig.26B, all nodes have been flooded and are present in the hierarchical queue. The processing of the nodes have to be red from bottom to top in each FIFO

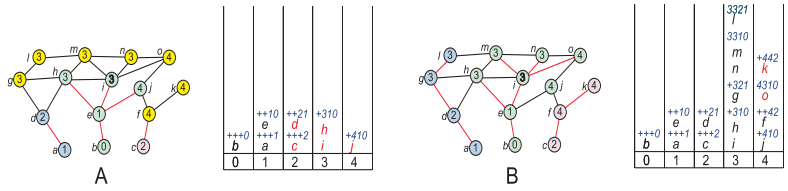

Fig. 26. 2 steps of flooding a node weighted graph with the status of the hierarchical queue.

and from left to right among the FIFOs. It clearly appears that the nodes have been processed in an increasing lexicographic order of infinite depth.

Should we entitle this paper "much ado about nothing", as it presents a theory and operators in order to achieve what the simplest watershed algorithm silently does since 1991 ?

Part V

The waterfall hierarchy and the emergence of minimum spanning trees and forests

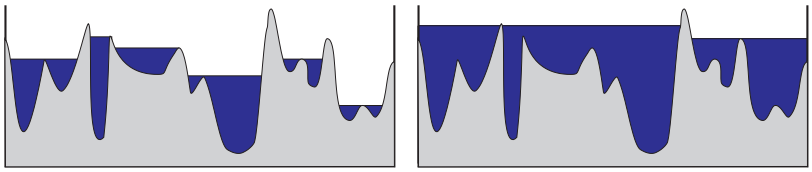

Fig. 27. Chaining the waterfall floodings

17 The hierarchy of nested catchment basins

17.1 Watershed and waterfalls

The waterfall flooding The waterfall hierarchy has been introduced by S.Beucher in order to obtain a multiscale segmentation of an image [4],[5],[5]. Given a topographic surface S_1, typically a gradient image of the image to segment, a first watershed transform produces a first partition π_1.

The waterfall flooding of S_1 consists in flooding each catchment basin up to its lowest neighboring pass point, producing like that a topographic surface with less minima. The watershed segmentation of this surface produces a coarser partition π_2 where each region is the union of a number of regions of π_1.

Chaining the waterfall floodings Flooding each catchment basin of S_1 up to its lowest pass point produces a new topographic surface S_2 which will be submitted to the same treatment as the initial surface S_1. Its watershed transform produces a second partition π_2. The partition π_2 is coarser than π_1 as each of its tile is a union of tiles of π_1.Fig.27 shows how the flooding of S_1 produces S_2, which, in the second figure, has also been flooded.

The same process can be repeated several times until a completely flat surface is created. The partitions obtained by the watershed construction on the successive waterfall floodings are coarser and coarser : they form a hierarchy.

The waterfall hierarchy A 1 dimensional topographic surface is represented through the altitude of its pass points in fig.28 below. The first flooding assigns weights to the nodes and its watershed construction produces 4 catchment basins, separated by 3 watershed lines. The second flooding has only 2 catchment basins separated by 1 watershed line. The last image orders the watershed lines of the initial image into 3 categories, in cyan the watershed lines which disappeared during the first flooding, in dark blue those which disappeared after the second flooding and in red the one which survived the second flooding.

Let us come back to the watershed on weighted graphs. The watershed of the topographic surface produces a partition π_1 into catchment basins, represented by its region adjacency graph RAG_1. The first flooding floods each catchment basin up to its lowest neighboring pass point. This corresponds to the erosion ε_{ne}

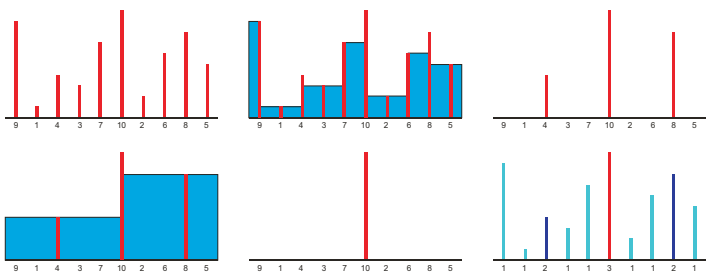

Fig. 28. The waterfall hierarchy

of the graph RAG_1. The theory of the watershed on weighted graphs can now be applied on this graph. The resulting watershed appears in form of minimum spanning forest MSF_1; each tree of the forest spans a catchment basin of the partition π_2.

The next level of the hierarchy may be represented by a new region adjacency graph RAG_2, whose nodes are obtained by contracting each tree of the forest MSF_1 in the graph RAG_1. Repeating the same treatment to the graph RAG_2 produces the next level of the hierarchy, where each tree of the minimum spanning forest MSF_2 has been obtained by merging several trees of the MSF_1. The process is then repeated until a graph is created with only one regional minimum.

Construction of the level 1 of the hierarchy We start with an arbitrary node or edge weighted and connected graph G. As explained earlier, we extract a graph flooding graph G'. For a steepness k, we prune G' and get $\downarrow^k G\prime$. The scissor operator χ produces a minimum spanning forest $F_1 = \chi \downarrow^k G\prime$, spanning the finest watershed partition, the lowest level of the hierarchy.

The graph representing the second level of the hierarchy is obtained by contracting all edges of the forest F_1 ; each tree becomes one node. The nodes are connected by edges of the graph G. The result of this contraction $G'_1 = \kappa(G, F_1)$. is again a connected graph, to which the same treatment as previously can be applied.

Construction of the level 2 of the hierarchy Only the nodes of the graph G'_1 are weighted : $G'_1 = (e_1, \diamond)$. The nodes will be weighted with ε_{ne} and we get the graph $(e_1, \varepsilon_{ne} e_1)$, which becomes a flooding graph of steepness k in $\downarrow^k (e_1, \varepsilon_{ne} e_1)$. A final scissor operator creates a forest F_2 spanning nodes of G'_1. F_2 represents the level 2 of the hierarchy, and the nodes of G spanned by each of its trees constitutes a catchment basin of level 2.

The contraction $\kappa(G'_1, F_1)$ produces again a connected graph $G'_2 = (e_1, \diamond)$ to which the same treatment may be applied.

This process is repeated until the graph $\kappa(G'_m, F_m)$ contains a unique regional minimum of edges.

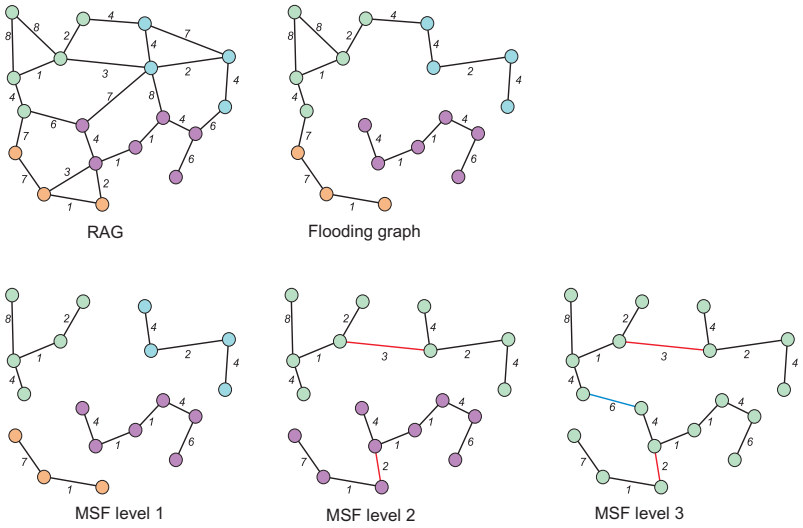

Fig. 29. An edge weighted graph, its floodign graph, and the minimum spanning forests at 3 waterfall hierarchical levels.

17.2 Emergence of a minimum spanning tree

Each new minimum spanning forest F_n makes use of new edges of the initial graph. The union of all MSF constructed up to the iteration m, $\bigcup_{k \leq m} F_k$, is a minimum spanning forest of the initial graph G, which converges to a MST of G as m increases. Hence the union of all edges present in the series F_n is a minimum spanning tree of the graph G.

The particular minimum spanning tree which emerges depends upon the depth k of the pruning operator $\downarrow^k$, and of the particular choices among alternative solutions made by the pruning operators χ used at each step. We call the operator extracting the minimum spanning tree $\mu(G)$.

Fig.29 in the next slide presents a RAG, its flooding graph in the first row, and in the second row the MSFs $\bigcup_{k \leq m} F_k$ for $m = 1, 2, 3$. The last one being the MST.

If G and G' are two flooding trees, G' being a partial tree of G included in G, then the pruning operators χ have less choices for pruning $G\prime$ as for pruning G. For this reason the family of MST derived for G' is included in the family of MST derived for G: $\{\mu(G')\} \subset \{\mu(G)\}$.

In particular if one considers the decreasing sequence of minimum spanning forests $\chi \downarrow^k G$, one obtains decreasing families of MST : for $k < l$, we have $\left\{\mu(\downarrow^l G)\right\} \subset \left\{\mu(\downarrow^k G)\right\}$. The choices are more and more constrained. One gets minimum spanning trees which are steeper and steeper for increasing k in $\downarrow^k G$.

17.3 The complete waterfall hierarchy in one run

Level by level construction of the hierarchy All methods presented so far produce a waterfall hierarchy step by step.

In terms of flooding, each catchment basins is filled up by a lake to its lowest pass point. Being not a regional minimm anymore, it merges with the catchment basins which is situated on the other side of this pass point. The resulting flooding is a new topographic surface which may be submitted to a new watefall flooding producing level 2 of the hierarchy and so on until a last flooding producing a completely flat surface.

In terms of graphs, we obtain a similar behaviour if two nodes i and ν_j separated by an edge e_{ij} merge every time e_{ij} is the lowest edge adjacent to either i or ν_j, or to both of them. Cutting all edges which are not the lowest edge for one of their extremities produces a forest, constituing the lowest level of the waterfall hierarchy.

The contraction of each tree of the forest into one node transforms the initial MST in a reduced MST, where each node represents a region of the level 1 of the waterfall hierarchy. The same process may then be applied to this new tree, and repeated as many times as necessary until the last level where only one region remains.

Construction of all levels in one run It is possible to construct the complete waterfall hierarchy in one run. The lowest level of the hierarchy is constituted by the nodes themselves. The nodes get the weight 0. After fusion of two nodes of weight 0 the new node belongs to a region of level 1 of the waterfall hierarchy. A region of level 1 agglomerates a number of isolated nodes and remains a region of level 1 of the hierarchy. Only the fusion of two nodes of level 1 produces a region of level 2. Again the region of level 2 may absorb a number of nodes of level 1 or 0 and remains at the same level. The fusion of 2 regions of level 2 produces a region of level 3, and so on.

The algorithm operates on the minimum spanning tree of the region adjacency graph. It processes the edges of the graph in increasing order. Each processed edge is assigned a new weight, indicating its rank in the waterfall hierarchy. It is then contracted, and both its extremities become a unique node, which is also labeled. By successive contractions, the tree is progressively reduced until there is only one node remaining. During the process, each edge of the tree gets its rank in the waterfall hierarchy.

Let T be the MST of the graph. All its nodes get a weight 0, and its edges keep their weight in the RAG. We process the edges in order of increasing weights. Suppose that the current edge to be processed is edge e_{ij} linking the nodes ν_i and ν_j with the weights $a(\nu_i)$ and $a(\nu_j)$. Remark that the nodes ν_i and ν_j may be the result of several preceding contractions of edges with lower weights. The edge e_{ij} is then contracted and gets a weight ρ_{ij} ; both nodes ν_i and ν_j become a unique node ν_k with a weight $a(\nu_k)$. The values of these weights is the following:

- if two nodes ν_i and ν_j belong to the same hierarchical level, i.e $a(\nu_i) = a(\nu_j)$, the node ν_k obtained through their fusion gets the hierarchical level $a(\nu_k) = a(\nu_i) + 1$ and the edge e_{ij} gets the same weight $\rho_{ij} = a(\nu_i) + 1$.
- If the hierarchy level of ν_i and ν_j is not the same, than the node ν_k obtained by their fusion keeps the hierarchical level of the highest node $a(\nu_k) = \max(a(\nu_i), a(\nu_j))$; whereas the edge e_{ij} gets the hierarchical level $\rho_{ij} = \min[a(\nu_i), a(\nu_j)] + 1$

Remark : in both cases, the waterfall weight ρ_{ij} of the edge e_{ij} may be expressed by the same formula: $\rho_{ij} = \min[a(\nu_i), a(\nu_j)] + 1$

Interpretation Consider again the modified Boruvka's algorithm, constructing level 1 of the waterfall hierarchy, applied on the MST T of the neighborhood graph. The nodes of the MST get a weight 0 and the edges are processed in increasing order. We want to construct a forest F.

Initially the forest F is empty. We treat the edges of the neighborhood graph in increasing order of their weights. Let e_{ij} be the current edge considered.

1. e_{ij} connects two isolated nodes ν_i and ν_j. e_{ij} is then the lowest neighboring edge of both ν_i and ν_j and is a regional minimum among the edges. The edge e_{ij} is introduced in the forest, that is, it gets a weight $\rho_{ij} = a(\nu_i) + 1 = 1$.
2. e_{ij} connects an isolated node ν_i with a subtree of T. Clearly e_{ij} is the lowest edge adjacent to ν_i, otherwise node ν_i would have been incorporated in a subtree earlier. The edge e_{ij} is introduced in the forest, that is, it gets a weight $\rho_{ij} = \min[a(\nu_i), a(\nu_j)] + 1 = 1$
3. e_{ij} connects two nodes ν_i and ν_j already joined to their respective subtrees through edges with lower weights than e_{ij}. Hence e_{ij} is not the lowest edge of the nodes ν_i and ν_j, and is discarded. The edge e_{ij} is not introduced in the forest ; it will be considered during the construction of later stages of the hierarchy.

This situation happens in the level by level watershed construction when two individual nodes are merged (event 1 in the modified algorithm of Boruvka) whatever the hierarchical level which is processed.

Illustration The construction of the waterfall hierarchy with the algorithm we just presented is illustrated in fig.30, where the topography of fig.28 is represented in form of a graph. The nodes represent the regions and appear as green dots. The red dots represent the edges. Initially all nodes have a valuation $a(i) = 0$. The valuation of the edges express the dissimilarity. The starting point of the algorithm is presented in line 1 of fig.30. Line 2 of fig.30 presents the treatment of the lowest edge with valuation 1. Its surrounding nodes in line 1 have valuations 0. Hence the valuation of this edge in the waterfall hierarchy is 1. The two adjacent nodes are contracted and the new node gets a valuation 1 ; here we cancel the second node and retain the first with valuation 1.

The treatment of the edge of weight 2 between lines 2 and 3 is similar. Idem for the edge with weight 3 between lines 3 and 4.

Lines 4 and 5 deal with a new situation : the edge with valuation 4 separates two nodes with valuation 1. As $a(i) = a(j)$, then the new weight of e_{ij} $\rho_{ij} = a(i)+1 = 2$ and the weight of the new contracted node v_{ij} is $a(v_{ij}) = a(i)+2$. The contraction between both nodes is represented by cancelling the second node. The successive lines of fig.30 present the complete treatment, which ends line 11 where all edges have been assigned their waterfall level.

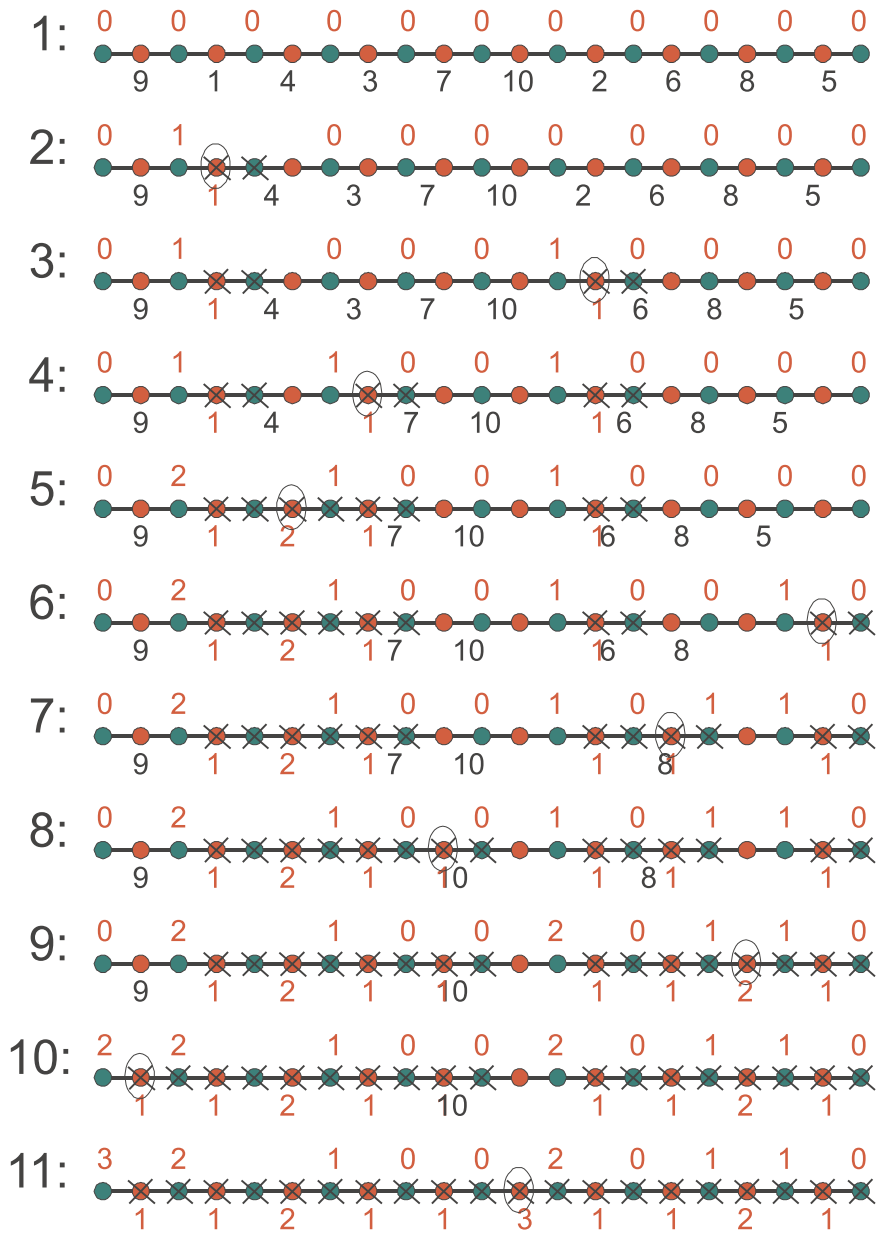

Fig. 30. Construction of the waterfall hierarchy in one run

Part VI

The watershed on images

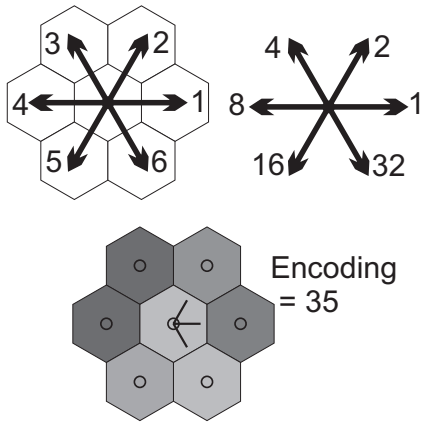

Fig. 31. The encoding of the directions of the neighbors in the hexagonal raster, and the weights of the corresponding bits in the binary representation of the arrows. An example with three arrows with weights 2, 1 and 32 is represented by the binary number 100011, i.e. the decimal number 35.

18 Representing an oriented graph for images.

In order to transpose to images the preceding algorithm initially defined for graphs, one has to find a representation of the drainage graph itself. The nodes are simply the pixels of the image to which the watershed algorithm is applied. The nodes hold three types of valuations and will be represented on three images. First, a gray tone image holding the initial distribution of gray tones and its evolution as the algorithm proceeds. The second image holds the labels of the minima and of the catchment basins as they expand. The last image is more original as it has to encode the drainage graph itself.

The grids on which images are represented have a regular structure, where each node has the same number of neighbors, in identical positions. Numbering the neighbors according to their direction allows to represent the neighborhood relation of each pixel with a binary number, where each bit encodes for one direction. The n-th bit is set to 1 if and only if there exists an arrow between the central point and its n-th neighbor ; such oriented arrows having as origin a node i are called *out-arrows* of i. Figure 31 shows the numbering of the directions in a hexagonal raster and the corresponding bit planes (on the right). The bottom image gives an example of encoding a particular neighborhood configuration : 2 + 1 + 32 = 35. This representation has been introduced by F. Maisonneuve in his seminal work on watersheds [30].

The algorithm creates and updates 3 images: 1) the gray tone image itself, 2) an image of labels representing the regional minima and the catchment basins in construction, 3) the encoding of the *out-arrows* representing the arcs of the drainage graph. Fig. 32 presents the three images. Fig. 32.1 represents the gray

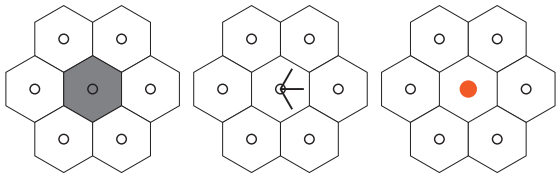

Fig. 32. The three images used for constructing the catchment basins : a gray tone image, an image of arrows and an image of labels.

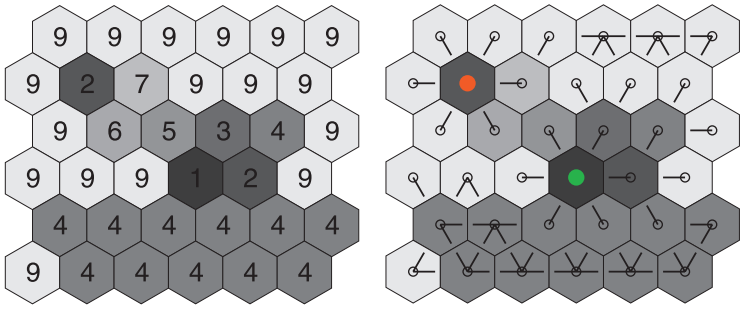

Fig. 33. A gray tone image, the arrows representing its drainage graph and the labels of the regional minima.

tone image. Fig. 32.2 represents the image of out-arrows encoding the drainage graph. Fig. 32.3 represents a label at the central position, represented as a colored dot.

Fig. 33.1 presents a gray tone image. Fig. 33.2 combines the three images: the gray tone value for each node, a colored dot representing the label of the minima and the initial *out-arrows* encoding the arcs.

19 An adaptive erosion and dilation, guided by the arrows

The successive prunings of arrows are easily implemented with two neighborhood transformations, the first being an adaptive erosion for propagating the gray tone values and pruning the arrows, the second being an adoptive dilation for propagating the labels of the regional minima as they are extended.

19.1 An adaptive erosion

We define an adaptive erosion and combined pruning on the arrowed image. It uses and updates both the gray tone image as the arrows image. If a pixel is without arrows, it is left unchanged. Otherwise, it is replaced by the lowest of

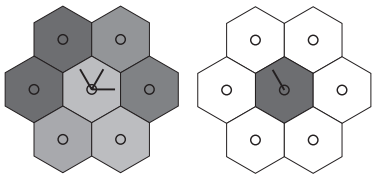

Fig. 34. An example of adaptive erosion and pruning.

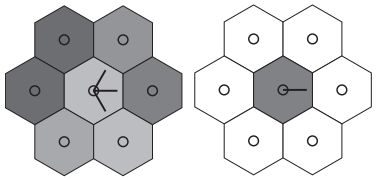

Fig. 35. Adaptive erosion and pruning

its arrowed neighbors and only the arrows towards these neighbors are kept. In figure 34 a pixel has two lowest neighbors, but only one of them is arrowed. Hence only the arrow towards this node is kept.

In figure 35 the two lowest neighbors are not arrowed : they are discarded and only the lowest arrowed neighbor is taken into consideration : its value is assigned to the central pixel and only the arrow towards it is kept.

When no arrow is present, the central pixel is left unchanged as in figure 36.

19.2 An adaptive dilation

The adaptive dilation propagates the labels of the regional minima. It modifies both the image of labels and of arrows. It is guided by the drainage graph.

Recall that the labels are represented by strictly positive values, the pixels without labels having the value 0 in the labeled image. Furthermore, the algorithm cares for the fact labeled pixels have no arrows. This is true at initialization, when the regional minima get their labels. It is also true during the

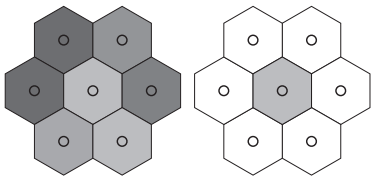

Fig. 36. In the absence of any arrow, the central pixel is left unchanged.

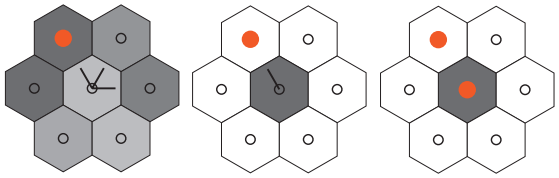

Fig. 37. Adaptative erosion, pruning and guided dilation of the labels

expansion of the catchment basins, as a pixel loses its arrows as soon it gets a label.

The label propagation is done by an adaptive dilation of the label images guided by the arrows image. One considers the pixels without labels (a pixel with a label has no out-arrows) ; such a pixel gets the highest label of its arrowed neighbors. In the absence of arrowed and labeled neighbors, this value is $0. The operation is thus an adaptive dilation$. Furthermore, every time a pixel gets a label, its arrows are suppressed, as it now belongs to a catchment basin. This produces an additional pruning of the drainage graph. Below we illustrate the combination of the adaptive erosion and dilation in a number of situations

Remark 8. 1) Neighboring nodes of the same catchment basins are not linked by an edge ; they are identified by the label they hold.
2) In case where a pixel without label has 2 or more out-arrows towards distinct labeled pixels, we have to chose one of them. With the highest of them, we have chosen an adaptive dilation. As a matter of fact, we may chose arbitrarily one of them. This is the only place where a choice takes place in the algorithm. It divides the catchment zones and produces a partition. Such situations are nevertheless rare, as we propagate the labels along the trajectories whose steepness is estimated by taking into account their total length. The necessity of a choice appears only in the case where two trajectories have exactly the same distribution of node weights, from top to bottom.

19.3 Combination of the adaptive erosion and dilation : illustration

The following figure shows how the adaptive erosion and dilation are used in sequence. The erosion assigns to the central pixel the value of its lowest arrowed neighbor ; only the arrow towards this neighbor is kept. And as this arrow points towards a labeled neighbor, this label is propagated to the central pixel by the adaptive dilation, and its arrows are suppressed. Figures 37,38,39,40 present how, in different neighborhood configurations, the combined adaptative erosion and dilation erode the gray tone, prune the arrows and propate the labels.

20 The complete watershed algorithm

Figure 41 shows that the algorithm can be initialized with arrows in all directions. After the first combined adaptive erosion and dilation, only the arrows towards

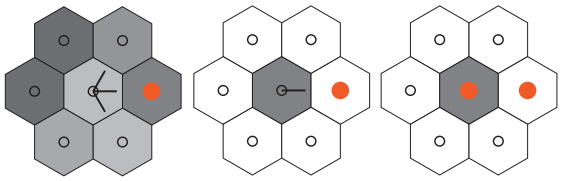

Fig. 38. Adaptative erosion, pruning and guided dilation of the labels

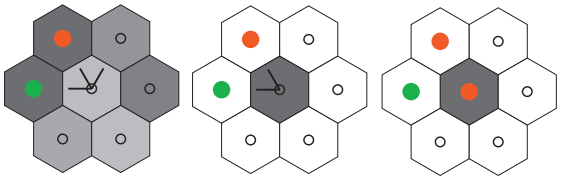

Fig. 39. In the case where two or more distinct labels are present in the direction of arrows, the highest of them, or one, chosen arbitrarily, is propagated

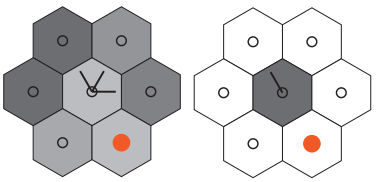

Fig. 40. Case where a labeled pixel is present in the neighborhood of the central pixel, but as it is not arrowed, it is not propagated to the central pixel.

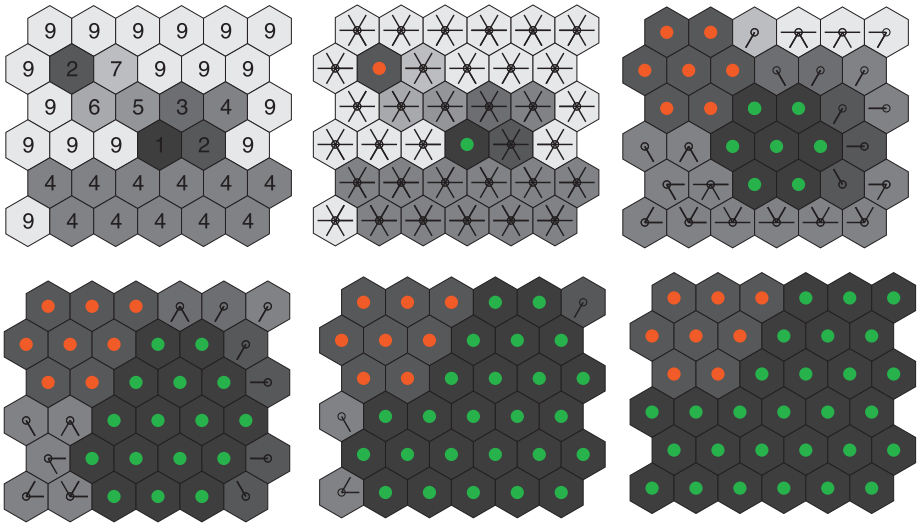

Fig. 41. Initial gray tone distribution, labeling of the regional minima and arrowing in all directions, followed by 4 iterations of a combined adaptive erosion and dilation.

the lowest neighbors are kept and the labels propagated accordingly. After the initialization phase, the combined erosion of gray tones, pruning of arrows and dilation of labels is iteratively applied until the labels cover the total domain. Convergence is attained after 4 iterations for figure 41.

By recording the arrows existing for each pixel, just before it gets its label, one obtains the final and steepest drainage graph, where all prunings have been done as illustrated in fig. 42. The trajectories of a drop of water falling on the surface are extremely selective and narrow. We will use this selectivity in the last part of the paper for following lines of steepest descent and thalweg lines on a topographic surface.

20.1 Successive steps of the pruning

During the successive adaptive erosions, no choice has ever been made. A choice may appear necessary when two when two distinct arrowed and labeled pixels are present in the neighborhood of a pixel. The adaptive dilation choses the highest of them. Other rules may be introduced, as for instance a random choice. The necessity of such choices is rare in natural images, as they only happen when two distinct drainage paths linking a node with two distinct minima has exactly the same distribution of weights.

20.2 Complexity

After the initial detection and labeling of the minima, the algorithm needs a number of iterations equal to the largest distance between a pixel in a catchment

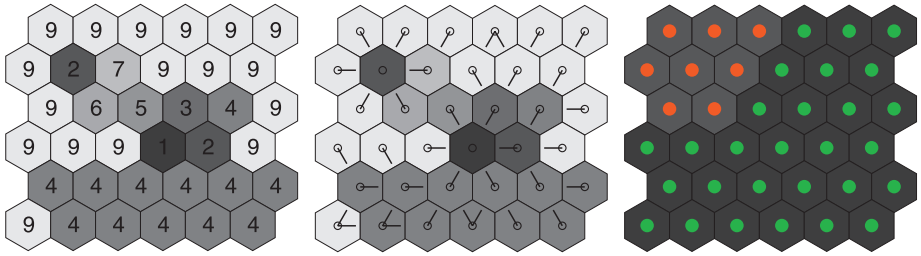

Fig. 42. Final catchment basins, and recording of the last arrow distribution of each node before it gets its label.

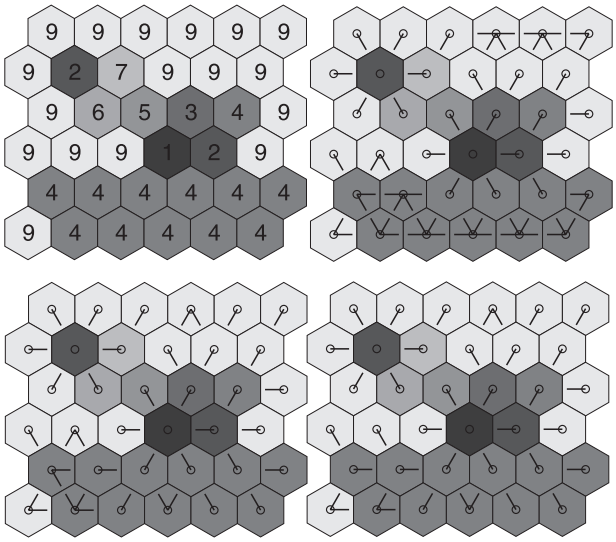

Fig. 43. Successive prunings of the arrows.

basin and its regional minimum. Each iteration consists in the combination of the adaptive erosion and dilation.

21 The problem of the plateaus

The plateaus pose a particular problem to all watershed algorithms which only consider local neighborhoods. As a matter of fact, a drop of water falling on a plateau has no clear direction for reaching the nearest regional minimum. The classical solution consists in constructing a geodesic distance transform to the lowest neighbors of the plateau and to follow the steepest descent line on this function. Fig. 44.1.1 shows on a topographic surface containing several plateaus with value 9. The geodesic distance within each plateau to its lower boundary is illustrated in fig. 44.1.2. This produces a topographic surface on which we may compute the drainage graph, as illustrated in fig. 44.2.2. By comparison, the steepest drainage graph produced by our algorithm is more selective and has less arrows It is illustrated in fig. 44.2.1.Furthermore no special treatment is required for dealing with the plateaus : they are treated as any other part of the topographic surface.

Remark 9. Some watershed algorithms use arrows for the construction of the watershed. F.Maisonneuve first assigned arrows to all pixels with lower neighbors, and then iteratively added out-arrows to pixels without arrows towards pixels with arrows. This produces the drainage graph of the geodesic distance on the plateaus [30]. F. Lemonnier, in a hardware implementation of the watershed, constructed separately the arrows of the drainage graph and those of the geodesic distance within the plateaus, before regrouping both and completing the watershed construction [29].

22 Illustration on a real image

The figures 45 and 46 illustrate the method on a real image. They contain in the top row a gray tone image and its gradient. The bottom row contains on the left the labeled regional minima and on the right the associated catchment basin. In fig. 45.2.2 they hold the same labels as the minima they contain. Fig. 46.2.2 is a mosaic image where each catchment basin takes the mean gray tone of the initial image in this region. Fig. 47 represents the final drainage graph and shows the image of the arrows encoded in false color.

23 Downstream trajectories of a drop of water

23.1 The algorithm for downstream propagation

We have successively presented two types of algorithms for constructing the catchment basins of a graph G. The first type prunes the graph to the outmost

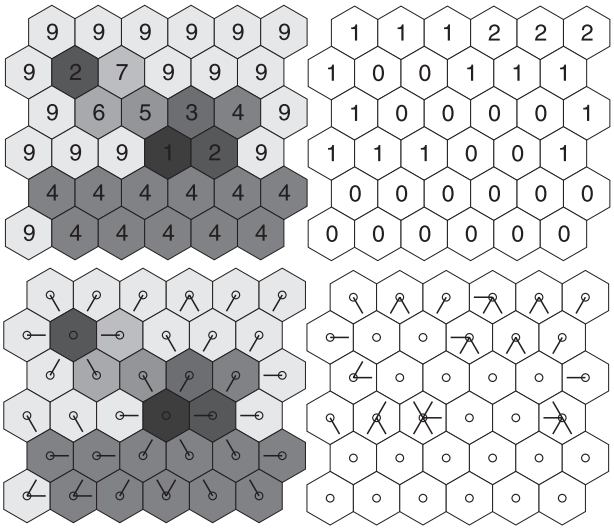

Fig. 44. Top: On the left a topographic surface and on the right the geodesic distance to its lower boundary on each plateau.
Bottom: On the left the arrows of the steepest drainage graph and on the right the arrows associated to the distance function.

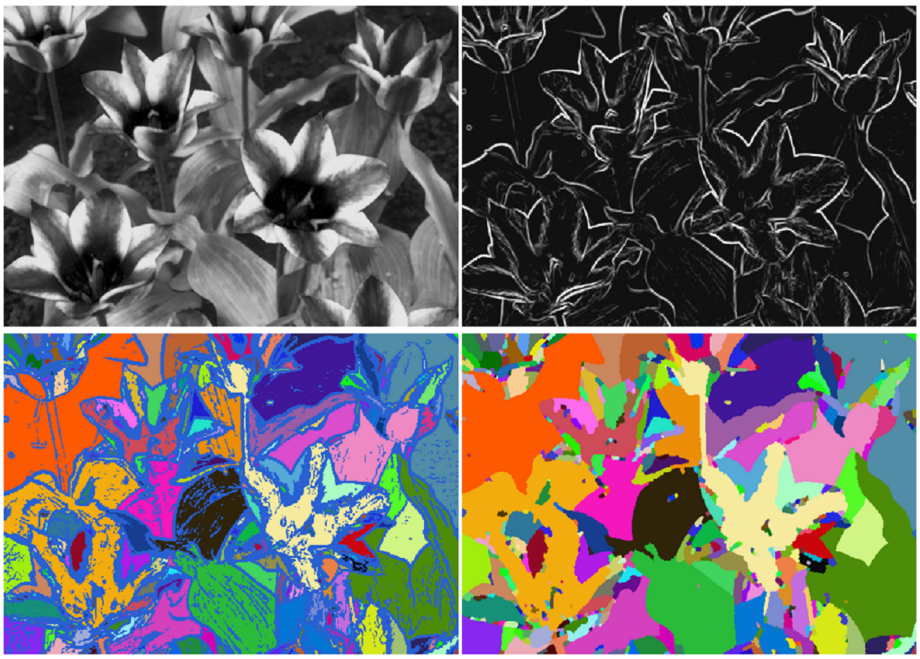

Fig. 45. Initial image, gradient, labeled regional minima and labeled catchment basins.

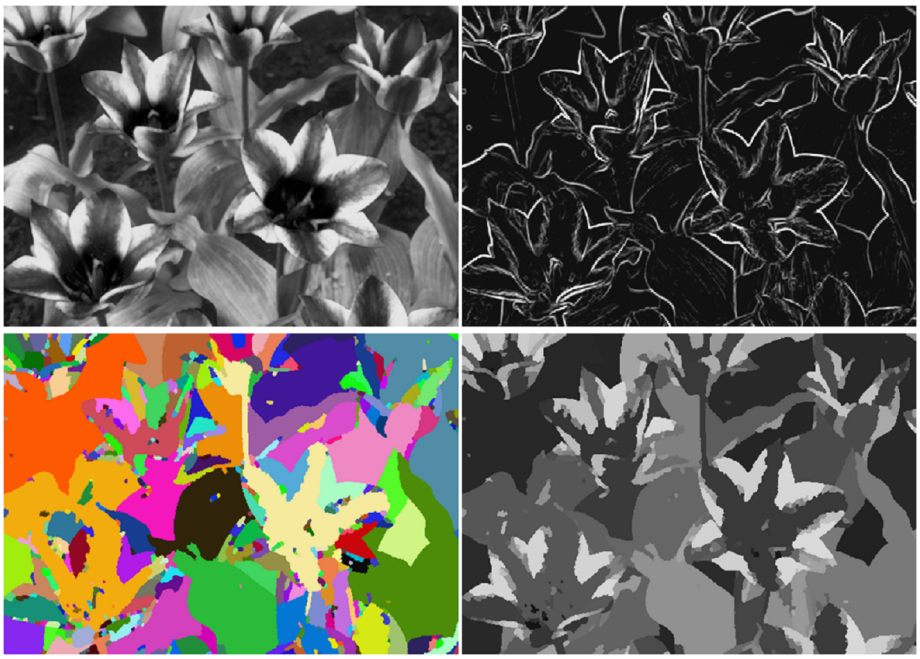

Fig. 46. Initial image, gradient, labeled regional minima and as final result the catchment basins containing each the mean gray tone of the initial image.

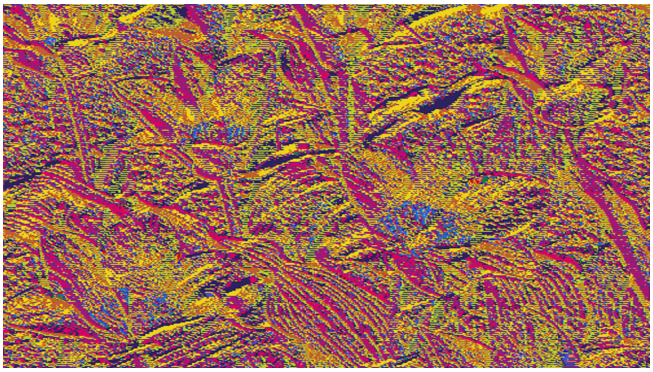

Fig. 47. The arrow image of the steepest flooding graph represented in false color.

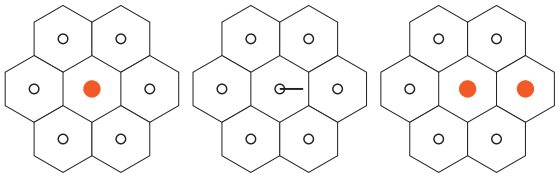

Fig. 48. Downstream following of a drop of water: a labeled node is expanded in the directions of its arrows to its neighboring nodes.

(pruning $\chi \downarrow^k G$), until only one adjacent edge remains for each node, leading along a steepest trajectory to a regional minimum. The result is a minimum spanning forest of maximal steepness.

The second flood the graph using a core expanding algorithm with infinite depth. Each node (the son) gets its label from a unique other node (the father), keeping only the edges linking father and son produces a minimum spanning forest also of maximal steepness. Both forests are included in the steepest graph $\downarrow^k G$; they may differ, if $\downarrow^k G$ contains nodes with two adjacent edges linking to different regional minima (this is unlikely on real images, as the geodesics leading to both minima have to have exactly the same lexicographic weight). In such a case, both the pruning χ as the core expanding algorithm may do different choices and assign such nodes to different catchment basins.

In both cases we obtain as a byproduct of the watershed construction this steepest forest, which may be used for following the steepest flooding trajectories. Given a few labeled starting points, it is possible to follow the downstream trajectories, simply by following the downstream arrows. Fig. 48 illustrates the operator which is used. If a pixel p has a label (fig. 48.1) and an arrow towards a neighboring pixel q, (fig. 48.2) then the label is propagated from p to q (fig. 48.3). If q has already a label, the maximum of both labels is chosen. This expansion is repeated until stability.

In fig. 49.1 two starting points are chosen, and assigned distinct labels, represented respectively by a red and a green dot. After downwards propagation following the arrows, both trajectories appear as labeled nodes in fig. 49.2, each of them reaching a regional minimum.

This method is applied on a DEM image in fig. 50.1 after the construction of the steepest drainage graph. A number of positions are chosen by hand in in fig. 50.2, where a drop of water will start its downwards trajectory, highlighted in the image on the right. The drop of water duly glides downwards until it meets and follows a river and ultimately reaches the boundary of the image in the direction of the sea, as illustrated in fig. 50.3. Each river keeps the label of its highest source and keeps this label unless it meets another label which is higher. The connected components colored in red and touching the boundary of the image represent the regional minima of the image.

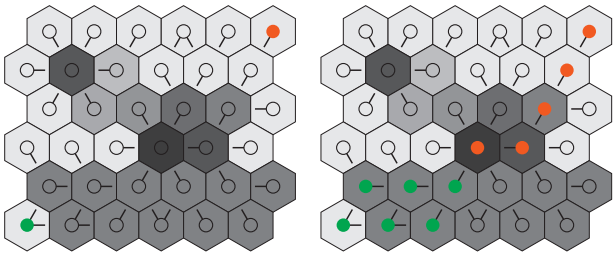

Fig. 49. Two starting nodes are chosen and assigned a label. On the right figure, the downwards trajectories are illustrated.

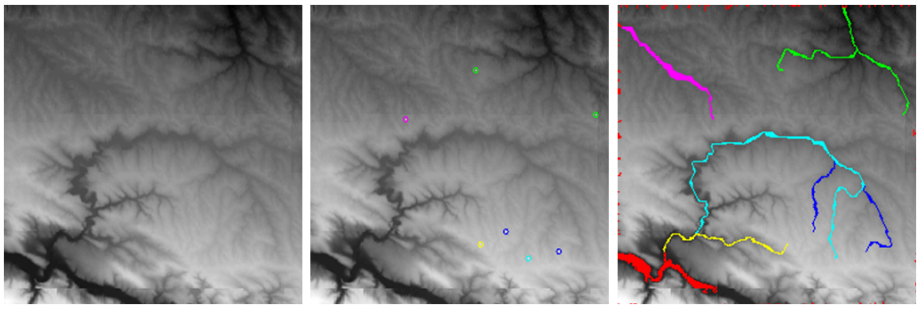

Fig. 50. After construction of the arrows of the DEM, a number of starting points are chosen and labeled (central image) and the downstream trajectories of a drop of water falling on these points highlighted in the right image.

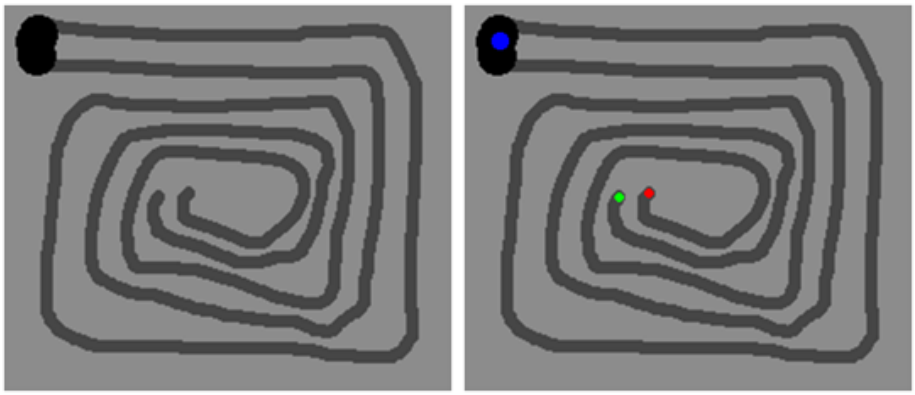

Fig. 51. Left : Two spirals on a bright background ending in a dark minimum
Right : A red and and a green dot mark the extremities of the spiral. A blue dot the minimum.

23.2 Application to the detection of fibers, cracks, thalweg lines.

Fig. 51 contains two spirals, intricated one in another, ending with a dark regional minimum. The background is brighter than the spiral. A red dot and a green dot have been put on the other extremity of both spirals. The regional minimum has been marked by a blue dot. The spirals are plateaus with a uniform gray-tone, with one lower boundary in the regional minimum. The steepest descent trajectories taking their origin in the red and green tots are represented in fig.52. They start from the red and green dots and follow the stripes until they reach the minimum

The spiral stripes in fig. 52 are plateaus of constant altitude, without internal structure for centering the trajectories, which appear at some places as large as the stripes in the right image. For a better centering of the trajectories, we have transformed the binary image into a gray tone image by constructing a geodesic distance to the lower borders of the plateaus. The arrows of the steepest drainage graph within a small square crossing a stripe are shown in fig. 53. The stripes are well centered and the trajectory along their thalweg well delineated. On this new relief, the detected trajectories are much thinner (fig.54)

23.3 Comparison with shortest path algorithms

The classical algorithms for following and highlighting thin and elongated dark structures (let us call it fiber) in a noisy bright background rely on shortest path algorithms (for instance cracks in a porous medium, hairs, glass fibers, vessels in 2 or 3 dimensions etc.). As the gray tone along the fiber is darker than in the background, the integral of gray tone along the fiber is smaller than along a path with the same length lying in the brighter background. This integral may be seen

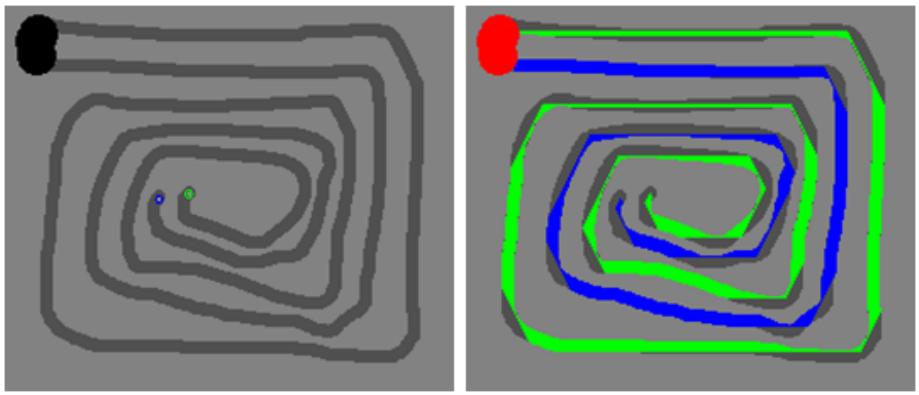

Fig. 52. The spiral stripes have a constant altitude, that is they are plateaus, without internal structure for centering the trajectories, which appear at some places as large as the stripes.

as a weighted distance transform. The method consists in computing the weighed distance along the fiber, first in one direction, then in the opposite direction ; the sum of both distance transforms is then minimal along the fiber. The method has been first proposed for detecting cracks in porous material [56],[60]. The method works well if the cracks or fibers or more or less rectilinear.

However, if the fiber is tortuous, like the spirals above, then a shortest path algorithm between the blue and the red dot or between the blue and the green dot will find shortcuts and will not follow the spiral. The classical solution consists in a stepwise progression : one progresses along the fiber on a short distance, so as to remain on it and initiates a new progression at the arrival point. Like that, little jump after little jump, one progresses. The length of the jump depends upon the contrast between fiber and background [17].

On the contrary, using the drainage graph does nor present this weakness : the trajectories are delineated and based on a long range lexicographic distance which makes it possible to follow them completely from the top downwards, until a regional minimum is met. Contrarily to the shortest path algorithm, the algorithm has not to be used stepwise, does not depend upon the contrast between foreground and background. It is also able to follow several fibers at the same time. However, it is certainly more sensitive to noise or missing data than the shortest path algorithms.

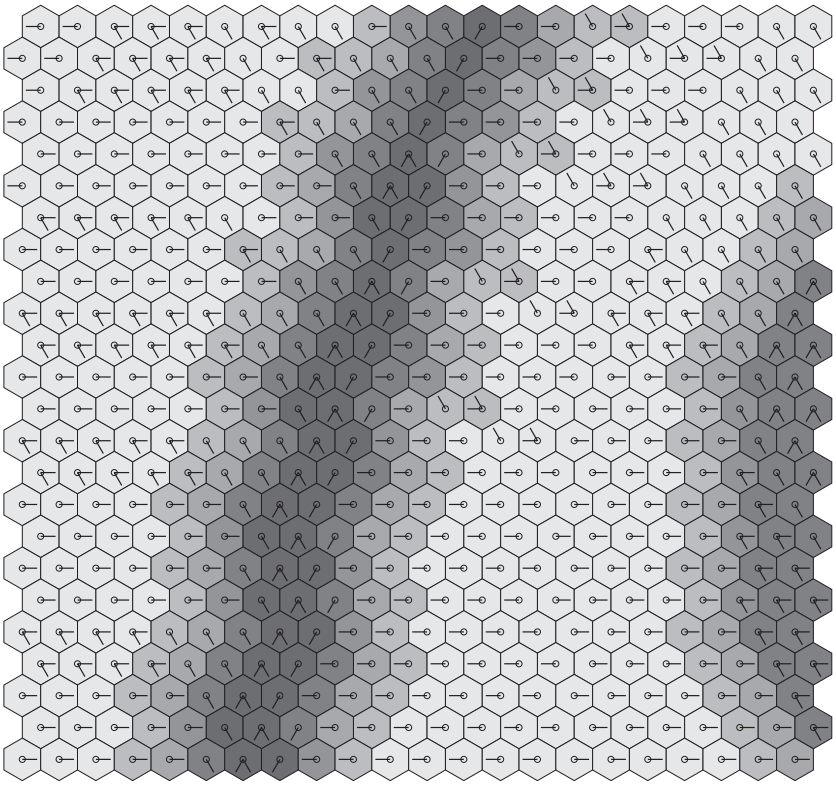

Fig. 53. The arrows of the steepest drainage graph after constructing the geodesic distance of each node of a plateau to the lower border of the plateau.

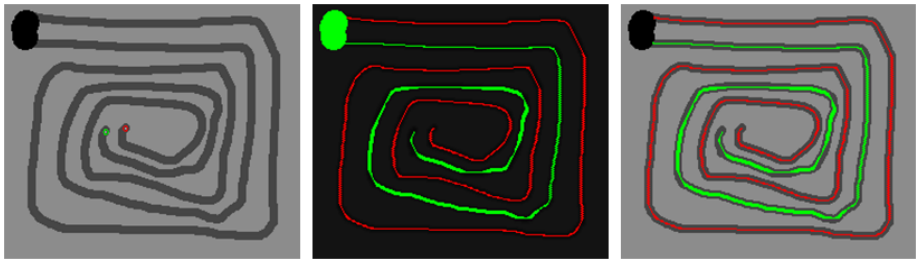

Fig. 54. Replacing each strip by a distance transform to its lowest boundary permits to center the downwards trajectories inside the stripes.

Part VII

The watershed by flooding

24 The catchment basins, skeletons by zone of influence for lexicographic distance functions

24.1 Drop of water gliding down or flood progressing upwards ?

Let us summarize what we learned in the first part of the paper. The operator $\downarrow^k G$ prunes the flooding graph and leaves only flooding tracks or paths with a steepness equal to k. From each node of the graph starts a NAP whose k first edges have a minimal lexicographic weight. If one follows such a path, the next k edges following each node as one goes downwards along the path also has a minimal lexicographic weight. This shows that each NAP is a geodesic line for a lexicographic distance function lexdist_k we define below.

In this second part of the paper we change perspectives. Whereas the first part considered a drop of water following a line of steepest descent we consider now a floooding scenario, where we start from the minima and construct skeletons by zone of influence for particular distance functions.

As a matter of fact, it is really only changing perspectives, as the lines of steepest descent remain the same, whether one descends them for reaching a regional minimum or whether one climbs them as a flooding would progress. What matters really in both situations is the steepness of the lines of steepest descent one considers. The steeper these lines, the rarer they are. The steepest lines having a given node as origin often are unique ; this node is then unambiguously ascribed to one catchment basin containing one regional minimum.

After the pruning $\downarrow^k$ some nodes belong to two or more catchment basins. In order to obtain a partition, one applies the scissor operator χ leaving for each node only one lowest adjacent edge. Like that the thick watershed zones are suppressed and the catchment basins form a partition.

It is possible to obtain the same result, if one labels the regional minima and computes for the other nodes the shortest lexicographic distance lexdist_k to the minima. If one applies a greedy algorithm, one may in addition propagate the labels of the minima all along the geodesics and construct a partition of the space. If a node is at the same lexicographic distance of two nodes, a greedy algorithm will arbitrarily assign to it one of the labels of the minima. This is quite similar to the operator χ which also does arbitrary choices.

24.2 Lexicographic distances of depth k

Defining a lexicographic distance along non ascending paths. Consider a flooding graph. In a flooding tracks, each node except the last one forms with the following edge a flooding pair, i.e. they have the same weights. And these weights are decreasing as one follows the flooding tracks downwards. The k first values, starting from the top, may be considered as a lexicographic distance of depth k. In what follows we give a precise definition of such distances.

The shortest distances and their geodesics may be computed with classical shortest paths algorithms. Propagating the labels of the minima along the

geodesics during their construction constructs the zones of influence of the minima, i.e a partition into catchment basins. The solution is not necessarily unique. However the number of solutions decreases with the depth of the lexicographic distance which is considered.

Comparing NAPs with the lexicographic order. Let S be the set of sequences of edge or node weights, i.e. elements of $\mathcal{T}$. Let S_k be the set of sequences with a maximal number k of elements. For a sequence $s \in S_k$, we define the lexicographic weight $w_k(s)$: it is equal to ∞ if s is not a NAP and equal to s itself otherwise.

We define an operator first_k which keeps the k first edges and nodes of any NAP, or the NAP completely if its length is smaller than k. The operator first_k maps any sequence of S into S_k.

We define on the NAPs of S_k the usual lexicographic order relation, which we will note $\prec$, such that: $(\lambda_1, \lambda_2, \ldots, \lambda_k) \prec (\mu_1, \mu_2, \ldots, \mu_k)$ if either $\lambda_1 < \mu_1$ or $\lambda_i = \mu_i$ until rank s and $\lambda_{s+1} < \mu_{s+1}$. We define $a \preceq b$ as $a \prec b$ or $a = b$.

Like that, it is possible to compare any two sequences s_1 and s_2 of S_k by comparing their weights:

- if s_1 and s_2 are not NAPs, then $w_k(s_1) = w_k(s_2) = \infty$ and we consider that $s_1 \equiv s_2$ (they are equivalent)
- if one of them, say s_1, is a NAP and not the other, then $w_k(s_1) \prec w_k(s_2) = \infty$ and $s_1 \prec s_2$
- if both of them are NAP, then they compare as their weights: $s_1 \prec s_2 \Leftrightarrow \{w_k(s_1) \prec w_k(s_2)\}$ and $s_1 \equiv s_2 \Leftrightarrow \{w_k(s_1) = w_k(s_2)\}$

If s_1 and s_2 are NAPs belonging to S, we compare them with: $s_1 \prec s_2 \Leftrightarrow w_k\left[\text{first}_k(s_1)\right] \prec w_k\left[\text{first}_k(s_2)\right]$

Defining an "addition" operator, comparing sequences For NAPs of S we define the operator $\boxplus_k$ called "addition", which operates as a minimum:

$$a \boxplus_k b = \begin{array}{l} a \text{ if } a \preceq b \\ b \text{ if } b \preceq a \end{array} \qquad \forall a, b \in S$$

The $\boxplus_k$ operation is associative, commutative and has a neutral element ∞ called the zero element: $a \boxplus_k \infty = \infty \boxplus_k a = a$

Defining a "multiply" operator, by concatenating sequences The operator $\boxtimes_k$, called "multiplication", permits to compute the lexicographic length $\Lambda(A)$ of a sequence, obtained by the concatenation of two sequences. Let $a = (\lambda_1, \lambda_2, \ldots, \lambda_k)$ and $b = (\mu_1, \mu_2, \ldots, \mu_k)$ we will define $a \boxtimes_k b$ by:

- if a or b is not a NAP then $a \boxtimes_k b = \infty$
- if a and b are NAPs and $\lambda_k < \mu_1$ then $a \boxtimes_k b = \infty$
- if a and b are NAPs and $\lambda_k \geq \mu_1$ then $a \boxtimes_k b = w_k\left[\text{first}_k(a \rhd b)\right]$, where $a \rhd b$ is the concatenation of both sequences.

In particular $a \boxtimes_k \infty = \infty \boxtimes_k a = \infty$. so that the zero element is an absorbing element for $\boxtimes_k$.

24.3 Algebraic shortest paths algorithms

A path algebra on a dioïd structure The operator $\boxtimes_k$ is associative and has a neutral element 0 called unit element: $a \boxtimes_k 0 = a$. The multiplication is distributive with respect to the addition both to the left and to the right.

The structure $(S, \boxplus_k, \boxtimes_k)$ forms a dioïd, on which Gondran and Minoux defined a path algebra [19], where shortest paths algorithms are expressed as solutions of linear systems.

Transposing the dioïd to square matrices Addition and multiplication of square matrices of size n derives from the laws $\boxplus$ and $\boxtimes_k$: for $A = (a_{ij})$, $B = (b_{ij})$, $i, j \in [1, n]$:

$$C = A \boxplus B = (c_{ij}) \Leftrightarrow c_{ij} = a_{ij} \boxplus b_{ij} \qquad \forall i, j$$

$$C = A \boxtimes_k B = (c_{ij}) \Leftrightarrow c_{ij} = \overset{\boxplus}{\sum_{1 \leq k \leq n}} a_{ik} \boxtimes_k b_{kj} \qquad \forall i, j$$

where $\overset{\boxplus}{\sum}$ is the sum relative to $\boxplus$.

As there is no ambiguity, for $\overset{\boxplus}{\sum_{1 \leq k \leq n}} a_{ik} \boxtimes_k b_{kj}$, we simply write $\sum_{1 \leq k \leq n} a_{ik} b_{kj}$

Unity and zero matrices With these two laws, the square matrices also become a dioïd with

zero matrix $\widehat{\varepsilon} = \begin{bmatrix} \varepsilon & \cdots & \varepsilon \\ \cdots & & \cdots \\ \cdots & & \cdots \\ \cdots & & \cdots \\ \cdots & & \cdots \\ \varepsilon & \cdots & \varepsilon \end{bmatrix}$

and unity matrix $E = \begin{bmatrix} e & \cdots & \varepsilon \\ \cdot\cdot & e & \cdots \\ \cdot\cdot\cdot\cdot & e & \cdots \\ \cdot\cdot\cdot\cdot\cdot\cdot & e & \cdots \\ \cdot\cdot\cdot\cdot\cdot\cdot\cdot\cdot & e & \cdot\cdot\cdot\cdot \\ \varepsilon & \cdots & e \end{bmatrix}$

We write $A^0 = E$

24.4 An algebraic approach to shortest paths

Lexicographic length of paths in a graph If $G = [X, U]$ is a weighted graph with

- a set X of nodes, numbered $i = 1, \ldots\ldots, N$.
- a set U of edges $u = (i, j)$ with weights $s_{ij} \in S$.

The incidence matrix $A = (a_{ij})$ of the graph is given by:

$$a_{ij} = \left\{ \begin{array}{ccc} s_{ij} & \text{si} & (i,j) \in U \\ & \varepsilon & \text{sinon} \end{array} \right\}$$

To each path $\mu = (i_1, i_2, ...i_k)$ of the graph, one associated its k-lexicographic weight $w(\mu) = s_{i_1 i_2} \boxtimes_k s_{i_2 i_3} \boxtimes_k .... \boxtimes_k s_{i_{k-1} i_k}$, which is different from ∞ only if the path is a NAP.

If π is a never increasing sequence, then $w_k\left[\text{first}_k(\pi)\right] = \text{first}_k(\pi)$

Shortest paths in the graph The shortest paths for the lexicographic distance between any couple of nodes may be computed thanks to A^n or $A^{(n)} = E \boxplus A^1 \boxplus A^2 \boxplus .....A^n$.

Lemma 12. $A_{ij}^n = \overset{\boxplus}{\underset{\mu \in C_{ij}^n}{\sum}} w(\mu)$ *and* $A_{ij}^{(n)} = \overset{\boxplus}{\underset{\mu \in C_{ij}^{(n)}}{\sum}} w(\mu)$

where:

- C_{ij}^n is the family of paths between i and j, containing $n+1$ nodes (not necessarily distinct). A_{ij}^n is then the shortest path of length n between the nodes i and j.
- $C_{ij}^{(n)}$ is the family of paths between i and j, containing at most $n+1$ nodes (not necessarily distinct). A_{ij}^n is then the shortest path of any length smaller than n or equal to n between the nodes i and j.

Defining $A' = E \boxplus A$, as $\boxplus$ est idempotent ($a \boxplus a = a \qquad \forall a \in S$), we have: $A'^n = A^{(n)}$

In a graph with N nodes, an elementary path has at most N nodes, separated by $N-1$ edges. Hence, necessarily $A^{(N)} = A^{(N-1)}$ and A^*, the limit of $A^{(n)}$ for increasing n, is also equal to $A^{(N-1)}$. A_{ij}^* is the shortest path of any length between i and j.

Thanks to A^*, the computation of the catchment basins is immediate. If m_1 is a regional minimum, then a node i belongs to its catchment basin if and only if $A_{im_1}^* \leq \overset{\boxplus}{\underset{m_j \neq m_1}{\sum}} A_{im_j}^*$

A^* may be obtained by the successive multiplications :

A, $A^2 = A \boxtimes_k A$, $A^4 = A^2 \boxtimes_k A^2, ...A^{2^i} = A^{2^{i-1}} \boxtimes_k A^{2^{i-1}}$ until $2^i \geq N-1$,, i.e. $i \geq \log(N-1)$

A^* verifies $A^* = E \boxplus A \boxtimes_k A^*$.

Multiplying by a matrix B: $A^*B = B \boxplus A \boxtimes_k A^*B$. Defining $Y = A^*B$ shows that A^*B is solution of the equation $Y = B \boxplus A \boxtimes_k Y$. Furthermore, it is the smallest solution.

With varying B it is possible to solve all types of shortest distances:

- The smallest solution of $Y = E \boxplus A \boxtimes_k Y$ yields $A^*E = A^*$

– with $B = \begin{bmatrix} \varepsilon \\ \vdots \\ 0 \\ \vdots \\ \varepsilon \end{bmatrix}$, one gets A^*B, the i-th column of the matrix A^*, that is the distance of all nodes to the node i.

24.5 Solving linear systems

Gondran and Minoux have shown that most of the classical algorithms solving systems of linear equations (Gauss, Gauss-Seidel, etc.) are still valid in this context and correspond often to known shortest paths algorithms defined on graphs. We now give a few examples.

The Jacobi algorithm Setting $B = \begin{bmatrix} 0 \\ \varepsilon \\ \vdots \\ \vdots \\ \varepsilon \end{bmatrix}$ and $Y^{(0)} = \begin{bmatrix} \varepsilon \\ \vdots \\ \varepsilon \\ \vdots \\ \varepsilon \end{bmatrix}$, the iteration $Y^{(k)} = A * Y^{(k-1)} \boxplus B$ converges to $Y^{(N)} = A^* * B$

The Gauss Seidel algorithm A is decomposed as $A = L \boxplus E \boxplus U$, where L is an inferior triangular matrix, E the unity matrix E, and U an upper triangular matrix. The upper part of L and lower part of U have the value ε.

The solution of $Y = A*Y \boxplus B$ is obtained by the iteration : $Y^{(k)} = LY^{(k-1)} \boxplus UY^{(k)} \boxplus B$

This algorithm is faster as Jacobi's algorithm, as the product $UY^{(k)}$ already uses intermediate results, freshly computed during the current iteration $Y^{(k)}$.

The Jordan algorithm The Jordan algorithm is used in classical linear algebra for inverting matrices, by successive pivoting. In our case, where the shortest paths are elementary path the algorithms is:

For k from 1 to N :

For each i and j from i to N, do: $a_{ij} = a_{ij} \boxplus a_{ik} * a_{kj}$

The greedy algorithm of Moore-Dijkstra Gondran established the algebraic counterpart of the famous shortest path algorithm by Moore-Dijkstra.

Theorem (Gondran): Let $\overline{Y} = A^*B$ be the solution of $Y = AY \boxplus B$, for an arbitrary matrix B. There exists then an index i_0 such that $\overline{y_{i_0}} = \overset{\boxplus}{\sum} b_i$.

The smallest b is such solution : $\overline{y_{i_0}} = b_{i_0}$

Each element of $Y = AY \boxplus B$ is computed by $y_k = \sum_{j \neq k}^{\boxplus} a_{kj} * y_j \boxplus b_k = \sum_{j \neq k, i_0}^{\boxplus} a_{kj} * y_j \boxplus a_{ki_0} y_{i_0} \boxplus b_k$

Suppressing the line and column of rank i_0 and taking for b the vector $b_k^{(1)} = a_{ki_0} y_{i_0} \boxplus b_k$, one gets a new system of size $N-1$ to solve.

The Moore Dijkstra algorithm can also directly be computed on a flooding graph G, as presented below.

Remark 10. We only presented a few of the classical algorithms for solving linear systems in the framework of dioïds. More of them may be found in [19]

25 Shortest paths algorithms on the graph

The algebraic approach to shortest paths algorithms due to Gondran et Minoux [19] is extremely elegant and powerful due to its wide applicability. In order to establish a junction with more classical shortest paths algorithms we now present shortest paths algorithms applied directly on the graph.

We start with the shortest path algorithm of Dijkstra [44], whose algebraic greedy formulation has been presented in the previous section. We

show that for a lexicographic distance of depth 1, it becomes algorithm for constructing the minimum spanning forest of Prim.

We then introduce "core expanding algorithms" which take advantage of the particular structure of the lexicographic distances. They are faster than the Dijkstra algorithm and better suited to hardware implementations. For a lexicographic distance of depth 2, the core expanding algorithm is rigorously identical with the classical watershed algorithm based on the topographic distance [47],[39].

25.1 Architecture of shortest path algorithms.

Algorithms for constructing the catchment basins which expand the regional minima have been introduced in the literature as flooding algorithms according the well known scenario : a topographic surface is flooded, the minima acting as sources of the flooding and dams are erected in order to prevent lakes originating from distinct minima to merge. All points flooded from the same minimum belong to the catchment basin of this minimum. We keep here this image of flooding, considering increasing distances from the minima as the altitude of the flood. This flooding is the usual flooding in the case of lexicographic distances of depth 1, as these distances are identical with the ultrametric distances (when applied to non increasing paths).

The shortest path or "flooding" algorithms below are applied on a quasi flooding graph, invariant by the opening γ_e (the regional minima may ad libitum be contracted beforehand). A domain D is used and expanded, containing at each stage of the algorithms the nodes for which the shortest distance to the minima is

known ; D contains the nodes which already have been flooded, that is, all nodes for which the shortest distance to the minima has been correctly estimated. . Initially the minima are labeled and put in D. We say that a flooding pair (j, l) is on the boundary of the domain D, if j is inside D and l outside D. In this case we say that "j floods l", or "l is flooded by j". We say that j belongs to the inside boundary $\partial^- D$ of D and l to the outside boundary $\partial^+ D$ of D.

The domain D is progressively expanded by progressive incorporations of nodes belonging to flooding pairs on the boundary of D.

We consider two types of algorithms. The first one is the classical algorithm by Moore and Dijkstra. At each moment, it estimates the shortest distance of the nodes belonging to $\partial^+ D$ and introduces the node with the smallest estimated distance into D. The second is of a different type. It considers the node with the smallest distance belonging to $\partial^- D$, and floods all its neighbors in $\partial^+ D$. The second is faster, as it is able to flood several nodes at once from a unique node in $\partial^- D$. This computation is made only once and the corresponding nodes are put into D. The algorithm of Moore-Dijkstra on the contrary needs to compute a new estimation of nodes in $\partial^+ D$ each time a node is introduced into D ; for this reason the distance of a given node may need to be estimated several times.

25.2 The Moore Dijkstra algorithm

Labeling the nodes The shortest path algorithm by Moore-Dijkstra constructs the distances of all nodes to the minima in a greedy manner. It uses a domain D which contains at each stage of the algorithm the nodes i for which the shortest distance $\delta_k^*(i)$ and label $\lambda(i)$ is known. For all nodes in $\partial^+ D$, this distance can only be estimated and is written $\delta_k(i)$

Initialisation:

The nodes of the regional minima are labeled and put in the domain D.

Repeat until the domain D contains all nodes:

For each flooding pair (j, l) on the boundary of D, estimate the shortest path as $\delta_k(l) = e_{lj} \boxtimes_k \delta_k^*(j) = \nu_l \boxtimes_k \delta_k^*(j)$

The node with the lowest estimate is correctly estimated. If the corresponding flooding pair is (j, l) do:

- $D = D \cup \{l\}$
- $\delta_k^*(l) = \delta_k(l)$
- $\lambda(l) = \lambda(j)$

Constructing a minimum spanning forest The Moore Dijkstra may also construct the minimum spanning forest spanning the catchment basins. Let F be the forest to be constructed.

Initialisation:

F is initialised with a minimum spanning tree spanning each or the subgraphs of the regional minima

Repeat until the forest F contains all nodes:

For each flooding pair (j,l) on the boundary of D, estimate the shortest path as $\delta_k(l) = e_{lj} \boxtimes_k \delta_k^*(j) = \nu_l \boxtimes_k \delta_k^*(j)$

The node with the lowest estimate is correctly estimated. If the corresponding flooding pair is (j,l) do:

- add the node l and the edge (j,l) to the forest F
- $\delta_k^*(l) = \delta_k(l)$

Remark 11. If needed, one may add the propagation of the labels for easily recognizing the catchment basins, thus combining both versions of the Moore Dijkstra algorithm.

Correctness of the Moore Dijkstra algorithm The node with the lowest estimate is correctly estimated and is introduced in the domain D. It is the extremity of a shortest path whose all other nodes belong to D ; it is necessarily the shortest path, as any other path would have to leave D through another boundary flooding pair with a higher estimate.

Controlling the Moore Dijkstra algorithm The Moore algorithm presented above correctly computes the shortest distances to the minima. However if several nodes with the lowest estimate coexist, the algorithm choses one at random. Such a random choice is problematic in the presence of plateaus, as they may be quite arbitrarily divided between neighboring catchment basins.

The Moore Dijkstra may be advantageously controlled by a hierarchical queue structure (see [36] for the description of a hierarchical queue). Each node, as it gets is estimate is put into the HQ. As long as the HQ is not empty, the extracted node is among the nodes with the lowest estimate, the one which has been introduced in the HQ first.

The HQ has thus the advantage to correctly sequence the treatment in the presence of plateaus: it treats the nodes in the plateaus from their lower boundary inwards. The processing order is proportional to the distance of each node to the lower boundary of the plateau.

The Moore Dijkstra algorithm : case where $k = 1$ The algorithms remains the same but the computations are simplified as $\delta_1(j) = \nu_j$ is simply the weight of the node j, which is the first node of a flooding path leading from j to a regional minimum. In other terms, the domain D (resp. the forest F) is expanded by introducing into D (resp. F) the flooding pair on its boundary with the lowest weight. In the case of F, this algorithm corresponds exactly to the algorithm of PRIM for constructing minimum spanning forests. The same algorithm has been used in [33] for constructing the waterfall hierarchy.

This is not surprising, as for $k = 1$, the lexicographic weight of a NAP simply is the weight of the first edge. The distance is in this case the ultrametric flooding distance.

Remark: There is a complete freedom in the choice of the flooding pairs which are introduced at any time into D. A huge number of solutions are compatible with this distance, some of them quite unexpected as illustrated by an example illustrated in Fig.22.

25.3 The core expanding shortest distance algorithm

The lexicographic distance of depth k of a node t is equal to the list of weights of the k first nodes of the steepest flooding path with t as origin. We define an operator extracting from a list λ_n of n elements the k first elements: $\lambda_k = \text{first}_k(\lambda_n)$. If $n \leq k$, then $\lambda_k = \lambda_n$.

Let us first construct the lexicographic distance of infinite depth. If the second node of the steepest flooding path of origin t is s, then $d(t) = \nu_t \boxtimes \delta(s)$. The node s is then the neighboring node of t for which $\delta(s)$ is minimal. The "core expanding shortest distance algorithms" take advantage from this remark. We first present the algorithm and then prove its correctness.

The core expanding algorithm of infinite depth The core expanding shortest path algorithm constructs the distances of all nodes to the minima in a greedy manner. It uses a domain D which contains at each stage of the algorithm the nodes i for which the shortest distance $\delta(i)$ and label $\lambda(i)$ is known.

Initialisation:

The nodes of the regional minima are labeled and put in the domain D.

Repeat until the domain D contains all nodes:

Among all nodes of $\partial^- D$ already flooded, take the one, say s, for which the distance δ is the smallest. For each neighboring node t of s inside $\partial^+ D$ do:

- $\delta(t) = \nu_t \boxtimes \delta(s)$.
- $D = D \cup \{t\}$
- $\lambda(t) = \lambda(s)$

The core expanding algorithm of depth k The preceding algorithm is easily adapted for lexicographic distance of depth k. One has to keep only the k first elements of the distances. This is easily achieved with the help of the operator first_k . Keeping only the k first elements has an advantage, manipulating shorter distances and a disadvantage, it is less precise.

Initialisation:

The nodes of the regional minima are labeled and put in the domain D.

Repeat until the domain D contains all nodes:

Among all nodes of $\partial^- D$ already flooded, take the one, say s, for which the distance $\text{first}_{k-1}[\delta_k(s)]$ is the smallest. For each neighboring node t of s inside $\partial^+ D$ do:

- $\delta_k^*(t) = \nu_t \boxtimes \text{first}_{k-1}[\delta_k(s)]$
- $D = D \cup \{t\}$

– $\lambda(t) = \lambda(s)$

Proof of the correctness of the algorithm

We first remark that the distances of the nodes t which are expanded by the core expanding algorithm are never decreasing. At each moment, one expands the node s with the smallest distance. And the nodes which are introduced have a distance which is higher than s, as their distance is equal to the concatenation of their weight and of the distance of s.

Consider a node t whose steepest flooding path is $\pi = (x_1 = t, x_2, ...x_n)$, the last node x_n belonging to a regional minimum. We suppose first that t has only one steepest path and examine later what happens if its has several of them. Let us show that the preceding algorithm correctly estimates the lexicographic distance of the node t. We will show by induction that the distances of all nodes x_i in the path π are corrrectly estimated.

The distance of x_n belonging to a regional minimum is correctly estimated at initialisation. Suppose that the distance of x_l is correctly estimated ; what about x_{l+1} ? The node x_l is put in the domain D at the moment its distance is estimated. As the algorithm proceeds, comes a moment when x_l has the smallest distance of all nodes in D. At this moment, the node x_{l+1} is still outside D ; otherwise it would have been introduced into D by expanding a node with a smaller weight than x_l. Expanding x_l assigns to x_{l+1} its correct weight.

By induction, we show that the weights of all nodes of π are correctly estimated.

Case where π is not unique : in the case where the node x_1 is the first node of several steepest flooding paths, these paths have the same series of weights. And at each iteration of the algorithm there are several nodes with the same minimal weight. The algorithm choses arbitrarily one of them. This has no incidence on the computation of the distance but on the propagation of the labels.

Controlling the core expanding shortest distance algorithm The core expanding shortest distance algorithm may also be advantageously controlled by a hierarchical queue structure. The hierarchical queue is advantageous, as it permits to introduce the nodes with varying weights and at the same time always extracts the one with the smallest weight, in order to expand it.

Each node, as it is introduced in D is put into the HQ. As long the HQ is not empty, the extracted node is among the nodes with the lowest estimate, the one which has been introduced in the HQ first. This node gives its label and the correct distances to all its neighbors outside D by which it is flooded.

A node introduced in the HQ earlier is closer to the lower border of the plateau than nodes introduced later.

In the case of lexicographic distances with infinite depth, one does not have to care for plateaus, as the lexicographic distance becomes higher as one enters deeper inside the plateaus, starting from the lower boundary.

The algorithm is particularly suitable for a hardware implementation: each node is entered in the HQ only once. On the contrary, with the Moore-Dijkstra

algorithm a node may be introduced several times in the HQ, as its estimate may vary before it gets its definitive value.

The core expanding algorithms for the construction of minimum spanning forests Each of the preceding algorithms puts the accent on the propagation of labels. The catchment basins are identified by their labels. One may in addition or alternatively put the accent on the minimum spanning forests spanning the catchment basins. The modified algorith is straightforward. Let us take for instance the case of infinite depth, the others being quite similar.

Initialisation:

A spanning forest F is constructed containing a minimumum spanning tree of each regional minimum subgraph.

Repeat until the domain D contains all nodes:

Among all nodes of F having neighboring nodes outside F, take the one, say s, for which the distance δ is the smallest. For each neighboring node t of s outside F do:

- $\delta(t) = \nu_t \boxtimes \delta(s)$.
- Add the node t and the edge (s,t) to the tree F.

Remark 12. If needed, one may combine both versions of the algorithm constructing the forest and labeling the regional minima.

The core expanding shortest distance algorithm: case where $k = 1$ For $k = 1$, $\delta_1^*(s) = \nu_s$.

Initialisation:

The nodes of the regional minima are labeled and put in the domain D.

Repeat until the domain D contains all nodes:

Among all nodes of $\partial^- D$ already flooded, take the one, say s, for which the distance $\mathrm{first}_0[\delta_1(s)] = \infty$ is the smallest. This means that we are able to expand all nodes of $\partial^- D$ whatever their value. However, as we have no order relation between them, we have to check for each node s of $\partial^- D$ which is expanded, whether the node t which is considered forms with s a flooding track, that is $\nu_t \geq \nu_s$.

For each neighboring node t of s inside $\partial^+ D$ and verifying $\nu_t \geq \nu_s$ do:

- $\delta_1(t) = \nu_t \boxtimes \mathrm{first}_0(s) = \nu_t$
- $D = D \cup \{t\}$
- $\lambda(t) = \lambda(s)$

This means that any node i in D can be expanded and may introduce in D each of the outside nodes, say j such that $i \leq j$. The node j and the edge (i,j) form a flooding pair. We have seen earlier, that the flooding couples form a minimum spanning forest of a graph in which all regional minima have been contracted ; as the lexicographic distance is the shortest possible, the minimum spanning forest has a steepness equal to 0. Fig.22 shows that such minimum

spanning forests with 0 steepness may take unexpected shapes, as there is maximum freedom for constructing them. This lexicographic distance of depth 1 is identical with the ultrametric flooding distance applied to lists of non increasing weights.

The core expanding shortest distance algorithm: case where $k = 2$ If $k = 2$, then $\delta_2^*(s)$ is the valuation of the node s and of its lowest neighboring node. And $\delta_1^*(s)$ is the valuation of the node s itself. The algorithm introduces more constraints as the previous one for $k = 1$. It becomes:

Initialisation : Introduce all regional minima in the domain D and label them.

Repeat until all nodes get a label:

Among all nodes of D flooded by a node outside D, take the one with the smallest weight. Let t be this node, inside D and s its flooding neighbor outside D.

Do :

- $D = D \cup \{s\}$
- $\lambda(s) = \lambda(t)$

If in addition, one uses a HQ for controlling the process, we get the classical algorithm for constructing catchment basins [36].

Initialisation: Label the minima and introduce their nodes into a HQ, each with a priority equal to its weight.

Repeat until all nodes get a label: As long as the HQ is not empty, extract the node j with the highest priority from the HQ:

For each unlabeled neighboring (on the flooding graph) node i of j :

* $label(i) = label(j)$
* put i in the queue with priority ν_i

As a matter of fact, this algorithm has appeared in another context, defining the catchment basins as the zones of influence of the minima for the topographic distance [39].

Reminder on the topographic distance Consider an arbitrary path $\pi = (x_1, x_2, ..., x_p)$ between two nodes x_1 and x_n. The weight ν_p at node x_p can be written:

$\nu_p = \nu_p - \nu_{p-1} + \nu_{p-1} - \nu_{p-2} + \nu_{p-2} - \nu_{p-3} + ... + \nu_2 - \nu_1 + \nu_1$

The node $k-1$ is not necessarily the lowest node of node k, therefore $\nu_{k-1} \geq \varepsilon_n \nu_k$ and $\nu_k - \nu_{k-1} \leq \nu_k - \varepsilon_n \nu_k$.

Replacing each increment $\nu_k - \nu_{k-1}$ by $\nu_k - \varepsilon_n \nu_k$ will produce a sum $\nu_p - \varepsilon_n \nu_p + \nu_{p-1} - \varepsilon_n \nu_{p-1} + ... + \nu_2 - \varepsilon_n \nu_2 + \nu_1$ which is larger than ν_p. It is called the topographic length of the path $\pi = (x_1, x_2, ..., x_p)$. The path with the shortest topographic length between two nodes is called the topographic distance between these nodes [47],[39]. It will only be equal to ν_p in the case where the path $(x_1, x_2, ..., x_p)$ precisely is a path of steepest descent, from each node to its lowest

neighbor. In other terms the topographic distance and the distance lexdist_2 have the same geodesics.

If we define the toll to pay along a path $\pi = (x_1, x_2, ..., x_p)$ as ν_1 for the first node and $\nu_i - \varepsilon_n \nu_i$ for the others, then the lowest toll distance for a node x_p to a regional minimum will be ν_p if there exists a path of steepest descent from x_p to x_1. In other terms, x_p and x_1 belong to the same catchment basins if one considers the topographic distance. But they also belong to the same catchment basin if one considers the depth 2 lexicographic distance, as, by construction, x_k is the lowest neighbor of x_{k-1}.

As a matter of fact, the preceding algorithm may now be reformulated as a cheapest path algorithm, expressed on the nodes of the graph. The weights are assigned to the nodes and not to the edges. Each node may be considered as a town where a toll has to be paid. The cost of a path is equal to the sum of the tolls to be paid in all towns encountered along the path (including or not one or both ends).

Initialisation: Label the minima and introduce their nodes into a HQ, each with a priority equal to its weight.

Repeat until all nodes get a label: As long as the HQ is not empty, extract the node j with the highest priority from the HQ, that is with the lowest cost ν_i:

For each unlabeled neighboring (on the flooding graph) node i of j :

* $label(i) = label(j)$

* put i in the queue with priority $\nu_j + \nu_i - \varepsilon_n \nu_i$. But, since j is the first neighbor of i coming out of the queue, hence it is the lowest neighbor, that means that $\varepsilon_n \nu_i = \nu_j$ and $\nu_j + \nu_i - \varepsilon_n \nu_i = \nu_i$.

Finally, any image f may be considered as the global toll of its pixel graph if one takes:

- as reference nodes the regional minima of the image. Each of them has as toll its altitude.
- as local toll for all other nodes, the difference between their altitude and the altitude or their lowest neighbor: $g = f - \varepsilon f$

If in addition, we give a different label to each regional minimum, we may as previously propagate this label along each smallest toll path. We obtain like that a tessellation: to each minimum is ascribed a catchment basin: the set of all nodes which are closer to this minimum then to any other minimum. Fig.55 presents a graph where each node has two weights, the left value representing the lower gradient of the right value, that is the difference between the right value and the right value of the lowest neighbor.

Fig.56 illustrates how the cheapest path algorithm reconstructs the values of the initial grey tones and at the same time propagates the labels of the minima.

As a conclusion the catchment basins of a node weighted graph are the SKIZ of the minima, both for the topographic distance and for the depth 2 lexicographic distance. Each node is the extremity of a geodesic line, which is the same both for the toll distance and for the lexicographic distance of depth 2.

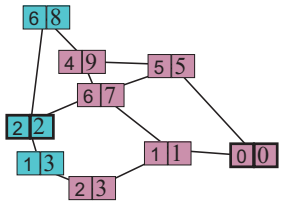

Fig. 55. Node weighted graph in which the left value represents the lower gradient of the right value, that is the difference between the right value and the right value of the lowest neighbor.

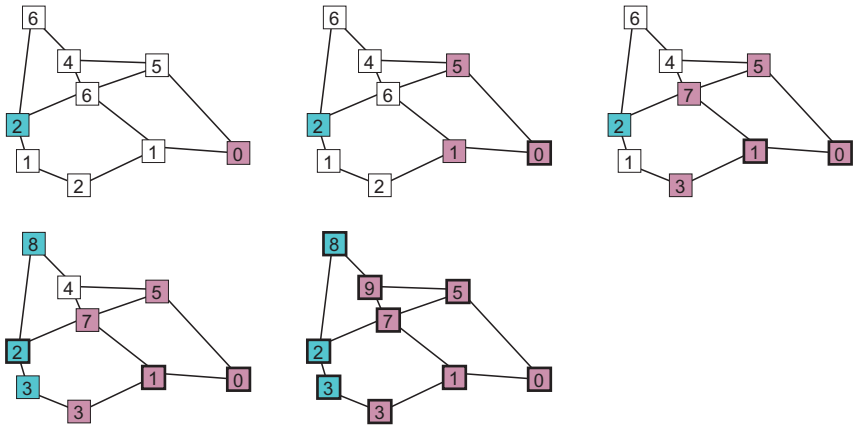

Fig. 56. Successive steps of the cheapest path algorithm

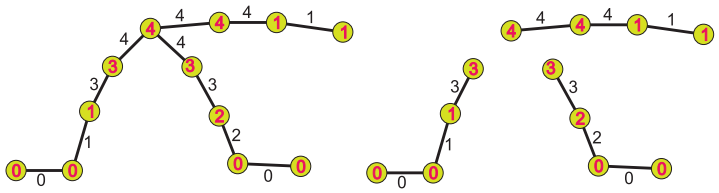

Fig. 57. Lexicographic distance of depth 1.

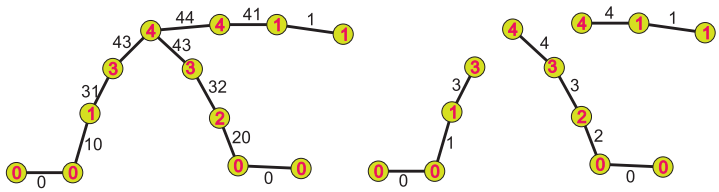

Fig. 58. Lexicographic distance of depth 2.

25.4 Top down or bottom up : the influence of the depth k

For increasing values of k, the k-lexicographic distance becomes more and more selective. At each step of the core expanding algorithm the number of nodes which are expandable also gets smaller and smaller. For this reason, the number of equivalent catchment basins compatible with a given lexicographic depth is reduced as the value of k becomes bigger.

This is in accordance with the fact that with increasing values of k, the pruning $\downarrow^k$ becomes more and more severe. After applying the operator $\downarrow^k$ on a flooding graph G, there remain only flooding tracks with k steepness. The same is true if we apply the operator ζ^k in order to prune edges of G and subsequently restore the initial weights of the edges onto the remaining edges.

We compare and illustrate on one hand the bottom up approach, applying a shortest path algorithm for lexicographic distances of increasing depth, on the other hand the top-down approach by pruning.

Bottom up : shortest lexicographic distances Figures 57, 58, 59 show respectively the shortest lexicographic distances of depth 1, 2 and 3. In each figure, the left one presents the distances and the right one presents a particular pruning χ, yielding a particular partition. The pruning of fig.59 is the only one applicable to the lexicographic distance of depth 3 but is also applicable to the lexicographic distances of depths 1 and 2. Similarly he pruning of fig.58 is also applicable to the lexicographic distance of depth 1. And the pruning of fig.57 is only applicable to distance 1. The number of solutions decreases with the lexicographic depth.

Top-down : successive prunings of a graph Repeating the operator $\zeta G = \downarrow \varepsilon G$ produces a decreasing series of partial graphs $\zeta^{(n)}G = \zeta\zeta^{(n-1)}G$, which are

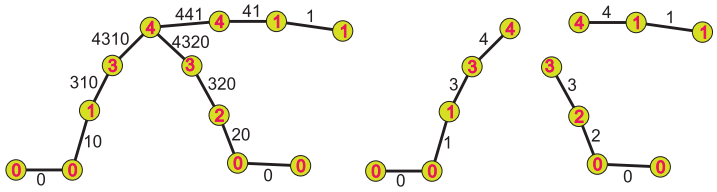

Fig. 59. Lexicographic distance of depth 3.

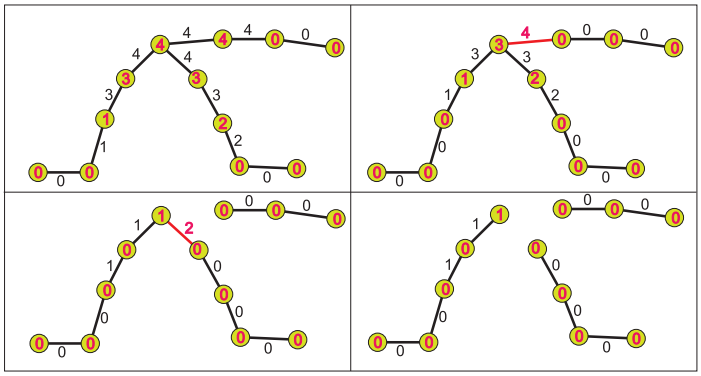

Fig. 60. Successive prunings of a graph when one applies the series $\zeta^{(n)}G$

steeper and steeper. In fig.60, we present in red the edge which is not the lowest edge of one of its extremities. After pruning this edge, the operator is applied again. Applying ζ a number n of times is equivalent to constructing the partitions compatible with a SKIZ for a lexicographic distance of depth $n+1$. In our case, $\zeta^{(3)}G$ produces a graph with the same edges as the pruning χ applied to the graph where each edge has been weighted by its lexicographic distance of depth 4 to the nearest minimum, illustrated in the previous figure.

Remark 13. If the only NAPs remaining in the graph have a k steepness, any shortest path algorithm with a lower steepness will extract them In particular the most simple algorithms for the distances d_1 or d_2 will extract catchment basins of steepness k.

25.5 Contrasts between flooding and pruning algorithms, illustrated by the watershed of a digital elevation map

Fig. 61.1.1 presents a digital elevation map of an existing landscape : each gray tone represents an altitude. Due to sensing errors, there are spurious regional minima in the topographic surface. As the rivers live the image in the direction of the see, the only minima which make sense are those on the boundary of the

Fig. 61. The gray tone image represents a digital elevation map. The central image shows all regional minima touching the boundary of the image. The right image presents the catchment basins.

image. A marker image is produced, equal to the relief on the boundary of the image and to ∞ elsewhere. The highest flooding of the relief under this mask has all its minima touching the boundary (see fig. 61.1.2); all other spurious minima due to sensing errors have been suppressed. lexicographic watershed produces the partition in catchment basins illustrated by fig. 61.1.3. Its catchments are the real catchment basins of the topographic surface, containing each a river, appearing as a thalweg line.

As announced earlier, the labels are propagated along lines of steepest descent ; at each iteration they progress by one step further. When they reach the top of a drainage path, they stop. This may be clearly seen in fig. 62 showing the extension of the catchment basins after 20, 40, 60, 80, 100 and 120 iterations of the elementary step of the algorithm (adaptive erosion of gray tones, dilation of labels and pruning of arrows). The construction of the catchment basins touching the lower boundary is achieved after a low number of iterations as they are small. They stop growing, independently of the adjacent catchment basins. The adjacent catchment basin, associated to the largest river reaches them after many more iterations.

This is in absolute contrast with the flooding algorithms which construct the catchment basins as attraction zones of the regional minima. For such algorithms, a catchment basin stops growing only when it arrives in contact with another catchment basin. Such algorithms are shortest distance algorithms and we presented earlier various implementation, either as the Moore Dijkstra algorithm or as the core expanding algorithms. In both cases, the propagation proceeds by increasing distances to the minima, which in the present context may be interpreted as increasing levels of flooding. The propagation of a flood front is only stopped when it meets another flood front.

The grows of catchment basins for both algorithms may now be compared. Fig. 62 shows the extension of the catchment basins after 20, 40, 60, 80, 100 and 120 iterations of the combined adaptive erosion of gray tones, pruning and dilation of labels. One remarks a striking difference with the classical algorithm for constructing the watershed which is based on the simulation of a flooding,

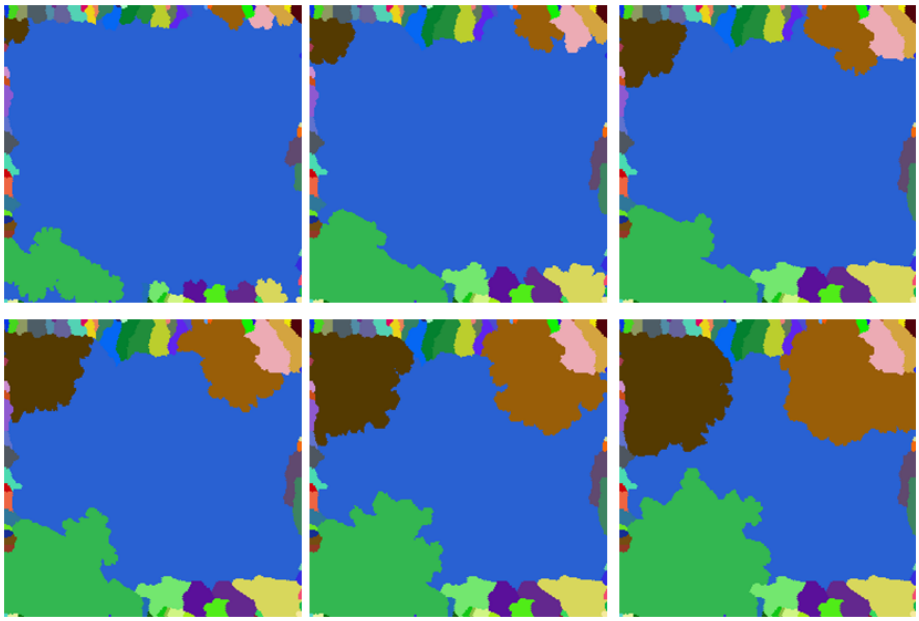

Fig. 62. Extension of the catchment basins after 20, 40, 60, 80, 100 and 120 iterations of the combined adaptive erosion of gray tones, pruning and dilation of labels.

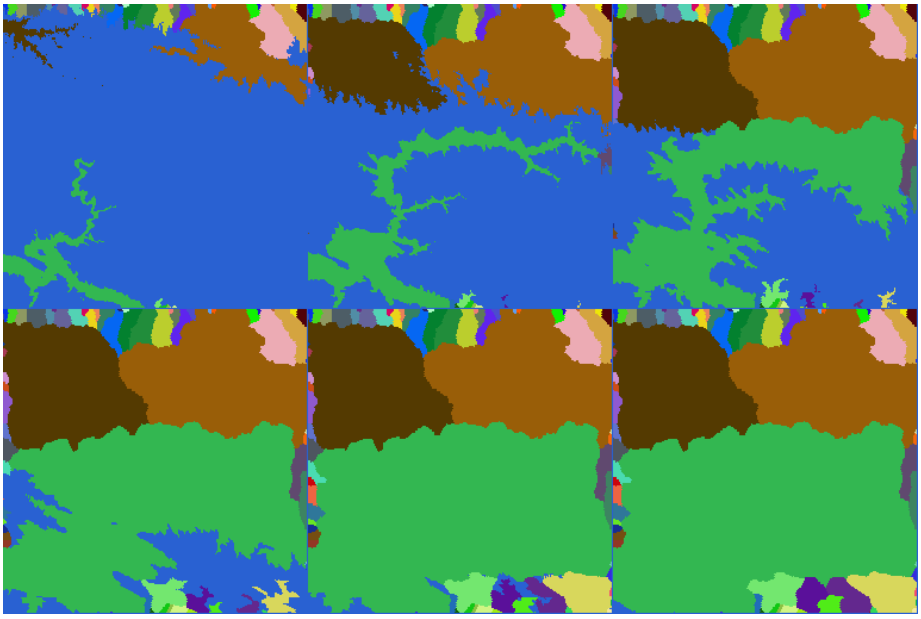

Fig. 63. Construction of the catchment basins with an algorithm based on uniform flooding

where a flood starting from the regional minima grows with a uniform altitude and progressively invades the topographic surface. Fig. 63 precisely shows the progressive construction of the catchment basin based on the flooding distance; the unflooded part appears in dark blue, the flooded parts appears as colored labels. The successive levels of flooding represented are 75, 105, 135, 170, 205 and 240 (gray-tones on a scale $[0, 255]$). The flooding algorithm is a greedy shortest distance algorithm based on the topographic distance. Each catchment basin stops growing everywhere it meets another catchment basin. With the steepest path algorithm on the contrary, the catchment basins are dilated at each iteration by a dilation of size one and they stop growing when they have reached their full extension, have they reached another catchment basin or not. As a consequence, if we suppress the label of a minimum at initialization, the catchment basin of this minimum will remain empty.

26 Chosing one watershed partition among how many ?

26.1 The choice of a particular watershed partition

The algorithms published in the literature start with a topographical surface or a weighted graph and associate to is a watershed partition. They all end up with a unique solution. In contrast, all along the present study we separated the

definition and characterization of a family of watershed partitions and the choice of a particular watershed partition within this family. As a matter of fact, as a catchment basin is the set of nodes linked with a regional minimum through a non ascending path, we associated families of watershed partitions with the steepness of the paths for constructing them. Increasing the steepness reduces the number of minima which is possible to reach from each node, and hence reduces the number of equivalent catchment basins.

The pruning and scissor algorithms This mechanism is clearly visible in the first part of the document, where the operators $\downarrow^k$ and ζ^k prune the graph, leaving only paths with a given steepness. As for $k > l$, the operator $\downarrow^k$ prunes more than the operator $\downarrow^l$, we indeed obtain a decreasing family of steepest paths and hence of associated catchment basins.

We then introduced an additional operator, the scissor operator χ, which randomly leaves only one flooding edge for each node. Alternatively, we defined a dilation which expands the regional minima step by step along the edges of the pruned graph ; there again, two distinct labels may meet in the central point, and the algorithm choses one at random.

The shortest distance algorithms, and the embedded choices. In the second part of the study, we select flooding paths as geodesics of lexicographic distance functions. As the lexicographic depth increases, there exist less and less geodesics. We proposed two families of algorithms producing as outcome a particular watershed partition. This means that the choice of a particular solution is embedded in the algorithm itself as analysed below.

If one considers the distance of a node to the nearest regional minimum as an altitude, one may consider the shortest paths algorithms as flooding algorithms, applied to a quasi flooding graph, invariant by the opening γ_e. A domain D is used and expanded, containing at each stage of the algorithms the nodes for which the shortest distance to the minima is known ; D contains the nodes which already have been flooded, that is, all nodes for which the shortest distance to the minima has been correctly estimated. . Initially the minima are labeled and put in D. We say that a flooding pair (j, l) is on the boundary of the domain D, if j is inside D and l outside D. In this case we say that "j floods l", or "l is flooded by j". We say that j belongs to the inside boundary $\partial^- D$ of D and l to the outside boundary $\partial^+ D$ of D.

The domain D is progressively expanded by progressive incorporations of nodes belonging to flooding pairs on the boundary of D.

We consider two types of algorithms. The first one is the classical algorithm by Moore and Dijkstra. At each moment, it estimates the shortest distance of the nodes belonging to $\partial^+ D$ and introduces one of the nodes with the smallest estimated distance into D. The second is of a different type. It considers one of the nodes with the smallest distance belonging to $\partial^- D$, and floods all its neighbors in $\partial^+ D$.

Each algorithm proceeds node by node ; at each step, one has to consider a family $\mathcal{F}$ of nodes whose value or estimated value is minimal ; for the Moore Dijkstra algorithm they are the nodes of $\partial^+ D$ with the smallest estimated distance. For the core expanding algorithm, they are the nodes with the smallest computed distance in $\partial^- D$. As long the domain D does not contain all nodes, the family $\mathcal{F}$ contains one or several equivalent nodes. If $\mathcal{F}$ contains several nodes, one of them has to be chosen by the algorithm : this is the place where a choice takes place.

If the choice is random, that is all possible choices may be considered, then the algorithms are able to construct all watershed partitions compatible with a given distance function. If on the contrary, not all choices are considered at each stage of the algorithm, then only a subset of all possible watershed partitions may be produced by the algorithm.

This may be of disadvantage, if the algorithm introduces a systematic bias in the choice of the solutions. It may be, on the contrary, an advantage, if the choices permit to select a watershed partition of quality. This will be the case if one uses for ordering the treatment of the nodes some particular data structure.

A number of algorithms have been published producing watershed partitions of quality. As there is no analysis on the respective role of the distance function and of the data structure, the reader attributes the merits of the algorithm to the clever definition of a distance function, whereas all the merits have to be ascribed to the data structure which has been used.

This discrete and obscure work of the data structures controling shortest distance algorithms is particularly spectacular when considering the hierarchical queue algorithms. Hierarchical queue algorithms have been introduced in order to gracefully treat the problem of the plateaus. But as we will see below, they do much more...

26.2 The problem of the plateaus in the shortest distance algorithms

The presence of plateaus of uniform altitude causes an obvious problem for the watershed algorithms. A drop of water falling inside a plateau has no means to orient itself ; in which direction should it flood ? This was not a problem for the first algorithms based on geodesic SKIZ of the minima inside the successive thresholds. Francis Maisonneuve [30] proposed to represent the slope of a topographic surface in terms of arrows from a node to its lower neighbors. For a drop of water falling on a plateau, he supposes that it isotropically extends itself until reaching a lower border, and that it will glide down the surface from this point. In other terms, a drop of water falling on a topographic surface will follow a line of steepest descent on the geodesic distance function of the plateaus to their lower border. Francis Maisonneuve proposed a propagation mechanisms to introduce additional arrows inside the plateau which provides the correct arrowing. Later F.Lemonnier explicitely constructed this geodesic distance function in order to be able to produce arrows in the direction of flooding everywhere [29].

As a matter of fact, the presence of plateaus introduces a 2-lexicographic order relation between nodes : a node p is higher than a node q if its altitude ν_p is higher than its altitude ν_q ; or if they both belong to a plateau, if p is farther then q from the lower border of the plateau.

A decisive progress was made by controlling the flooding with hierarchical queues [36]. This data structure is a series of prioritized FIFOs. It permits to enqueue each node p in the FIFO with the priority ν_p. The dequeue on the contrary consists in extracting the node which entered first into the FIFO with the highest priority. As the nodes close to the lower border of the plateau enter the queue before the inside nodes, the nodes inside the plateaus are indeed flooded in the 2-lexicographic order defined above. The algorithms based on hierarchical queues became the fastest algorithms for the construction of catchment basins.

As we have seen, many algorithms for the construction of watersheds are shortest path algorithms for various types of distances. Among them we highlighted the Dijkstra shortest path algorithm and presented the core expanding shortest paths algorithms which both are advantageously controlled by a hierarchical queue. In both cases, a domain D is used and expanded, containing at each stage of the algorithms the nodes for which the shortest distance to the minima is known ; D contains the nodes which already have been flooded, that is, all nodes for which the shortest distance to the minima has been correctly estimated. The domain D is progressively expanded by progressive incorporations of nodes belonging to flooding pairs on the boundary of D.

We consider two types of algorithms. The first one is the classical algorithm by Moore and Dijkstra. At each moment, it estimates the shortest distance of the nodes belonging to $\partial^+ D$ and introduces a node with the smallest estimated distance into D ; if there are several nodes with equivalent distances, any one of them may be chosen, yielding the same final result.

The second is of a different type. It considers the node with the smallest distance belonging to $\partial^- D$, and floods all its neighbors in $\partial^+ D$. Here again if there are several nodes with equivalent distances, any one of them may be chosen, yielding the same final result.

For both types of algorithms, one may use a "bucket" containing all nodes which could equivalently be chosen the algorithm at any time. The choice of the element to extract from the bucket being arbitrary or random. Doing that would produce correct distances, but being not further constrained, the watershed partitions possibly produced would be extremely numerous, and some of them would be odd looking (as for instance the figure &&&). Ordering the nodes to be processed with a hierarchical queue produces a unique solution. This means that among all solutions compatible with a given distance, the hierarchical queue favors one.

One may wonder, to which extend the family of resulting watershed partitions is reduced if one uses a hierarchical queue. Furthermore, is there not a risk that, due to the scanning order, the resulting watershed partition is systematically biased ? If this is the case how severely is it biased ?

In order to get a more clear idea on the subject we investigated more precisely how the hierarchical queue orders the treatment of the nodes.

The core expanding algorithm of depth 2 controlled by a hierarchical queue Consider the simple core expanding algorithm of depth 2 [36] presented earlier. It is described in algorithm 1.

Algorithm 1: The simple core expanding algorithm of depth 2

Input: A quasi flooding graph
A hierarchical queue

Result: The labeled nodes of the catchment basins

1 **Initialisation**: Label the minima and introduce their boundary nodes into a HQ, each with a priority equal to its weight.

2 **while** *HQ not empty* **do**
3 extract the node j with the highest priority from the HQ
for *each unlabeled neighboring node* i *of* j **do**
4 $label(i) = label(j)$
5 put i in the queue with priority ν_i

We will rephrase this algorithm and introduce tags into the queues as presented in algorithm 2. We will see that the nodes between two tags are equivalent and have the same lexicographic distance of infinite depth.

Illustration of the algorithm The preceding algorithm is better understood by contemplating fig.64 continued in fig.65, where a node weighted graph with three regional minima is progressively flooded. The hierarchical queue is a series of FIFO queues $(h0, h1, h2, ...)$. The queue $h0$ has the highest priority. Each time that a node i in the HQ is dequeued and expanded, each of its unlabeled neighbors is put in the queue and the edge linking both nodes gets a red color in fig.64. Alternatively, an arc could be created between each flooded node towards the node by which it has been flooded.

- A: the regional minima are labeled and their boundary nodes are put onto the hierarchical queue : each node is put in the queue with a priority equal to its weight. A tag is put in each non empty queue equal to the weight of the nodes in the queue. Each such tag represents the lexicographic distance to the closest regional minimum of the nodes below the tag, as these nodes belong to the regional minimum itself.
- B: the node b belongs to the queue with the highest priority 0 and is the only node below its tag equal to 0. This node is expanded (it takes the color blue in the queue) and its neighbor e with weight 1 is put in the queue $h1$

Algorithm 2: The core expanding algorithm of depth 2 with tags

Input: A quasi flooding graph
A hierarchical queue

Result: The labeled nodes of the catchment basins

Initialisation: Label the minima and introduce their boundary nodes into a HQ, each with a priority equal to its weight. Introduce a tag on top of the non empty hierarchical queues with the value equal to the weight of the nodes in the queue. Like that, the tag indeed indicates the lexicographic distance of the node to the regional minimum, as they belong to this minimum.

while *HQ not empty* **do**
  while *the next tag has not been reached in the highest priority queue* **do**
    extract the node j with the highest priority from the HQ
    for *each unlabeled neighboring node i of j* **do**
      $label(i) = label(j)$
      put i in the queue H_{ν_i} with priority ν_i
      set H_{ν_i} as active
  extract the tag in the highest priority queue; let τ be the value of this tag
  for *each active queue H_λ* **do**
    Introduce a tag equal to $\lambda \rhd \tau$ into the queue
    Set the queue as inactive

with a red color. A tag equal to $\nu_e \triangleright \text{tag}(b) = 1 \triangleright 0 = (10)$.
Similarly, the node a in $h1$ is expanded and its neighbor d is put into $h2$. A tag equal to $\nu_d \triangleright \text{tag}(a) = 2 \triangleright 1 = (21)$ is put into $h2$.

- C: The node e in $h1$ is expanded and its three neighbors put in the hierarchical queue. Two of its neighbors, h and i, with weight 3 are put into $h3$. A tag equal to $\nu_h \triangleright \text{tag}(e) = 3 \triangleright (10) = (310)$ is put into $h3$.
  The third neighbor, j with weight 4 is put into $h4$. A tag equal to $\nu_j \triangleright \text{tag}(e) = 4 \triangleright (10) = (410)$ is put into $h4$.
- D: The node c in $h2$ is expanded and its neighbor f is put into $h4$, followed by a tag equal to $\nu_f \triangleright \text{tag}(c) = 4 \triangleright 2 = (42)$ is put into $h4$.
- E: The node d in $h2$ is expanded and its neighbor g is put into $h3$, followed by a tag equal to $\nu_g \triangleright \text{tag}(d) = 3 \triangleright (21) = (321)$ is put into $h3$.
- F: the next package to expand contains the two nodes h and i, which are followed by the tag (310) in $h3$. The node i in $h3$ is expanded and its neighbors m and n are put into $h3$. The node h has no unlabeled neighbor to expand. Finally a tag equal to $\nu_m \triangleright \text{tag}(i) = 3 \triangleright (310) = (3310)$ is put into $h3$.
- G: The node g in $h3$ is expanded and its neighbor l is put into $h3$, followed by a tag equal to $\nu_l \triangleright \text{tag}(g) = 3 \triangleright (321) = (3321)$ is put into $h3$.
- H: The nores m, n, l and j are dequeued ; they have no unlabeled neighbor. Finally, the node f in $h4$ is expanded and its neighbor h is put into $h4$, followed by a tag equal to $\nu_k \triangleright \text{tag}(f) = 4 \triangleright (42) = (442)$ is put into $h4$. The next two nodes o and k may then be dequeued ; they have no unlabeled neighbors.

Analysis of the algorithm The tags present in the queue of priority λ all have as first element the weight λ, as they have been formed by the concatenation of λ and of a smaller tag. The tags are increasing from bottom to top in each queue and from lower priority queues to higher priority queues.

The plateaus are indeed treated correctly : the nodes g, h and i, at a distance 0 of the lower border, enter before the nodes n, m and l into the queue, at a distance 1 from a lower border. On the other hand the nodes h and i, with a lower lexicographic distance to a minimum, enter before g, l, m and n with a higher lexicographic distance to a minimum.

Each group of nodes between two tags has the same lexicographic distance to the closest regional minimum ; this distance is equal to the tag above them.

Obviously, the tags play no role in the ordering of the nodes in the HQ. The simple algorithm without tags and the algorithm with tags order the nodes in the same way.

Creation of the graph $\downarrow^\infty G$ Suppressing all edges through which does not pass a path of infinite steepness produces the graph $\downarrow^\infty G$. This same graph may be produced by a hierarchical queue flooding modified as follows.

We use the same graph as in fig.64. The outcome of the algorithm is the same up to the fig.64E, as at each iteration the input bucket has only one element. The modification is presented in fig.66. The input bucket to be expanded in fig.66E

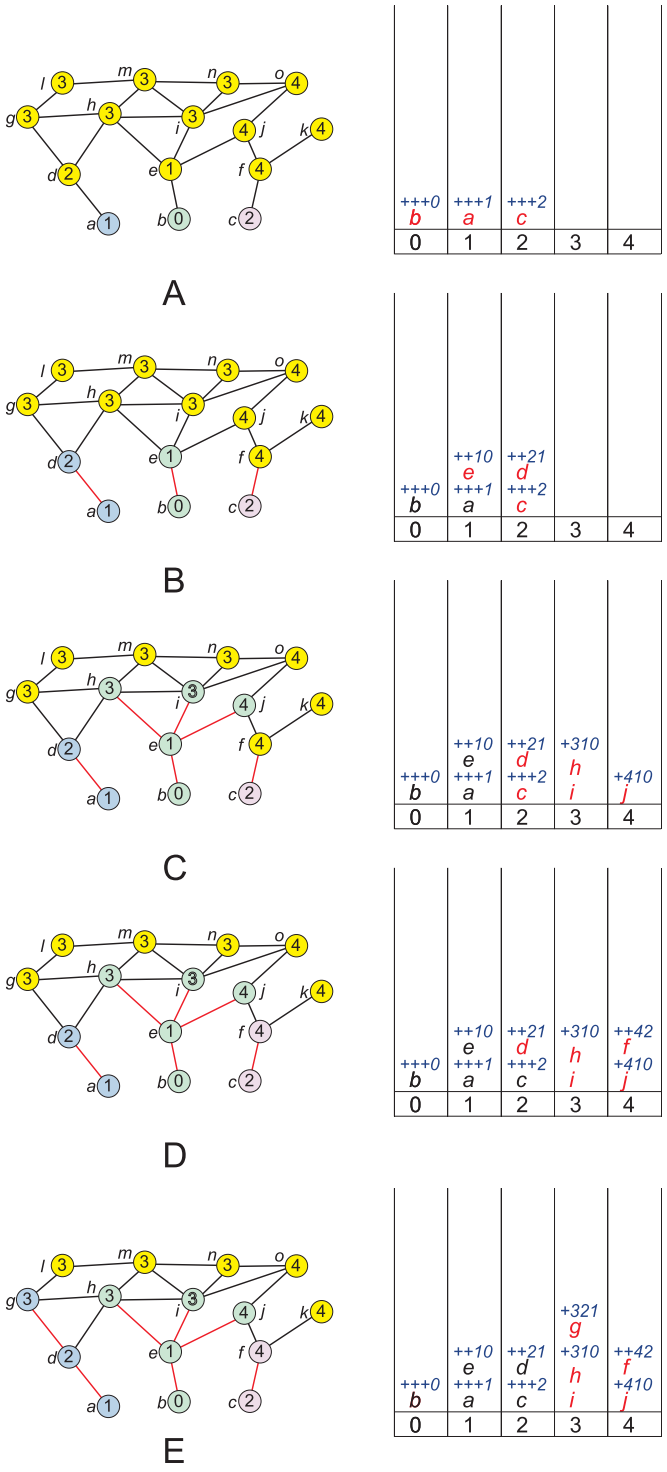

Fig. 64. Flooding from the minima of a node weighted graph controlled by a hierarchical queue. The nodes in the queue have two colors : the already expanded nodes are in dark blue and the nodes still to expand are in red. The lexicographic distance of each package of nodes is placed above the package.

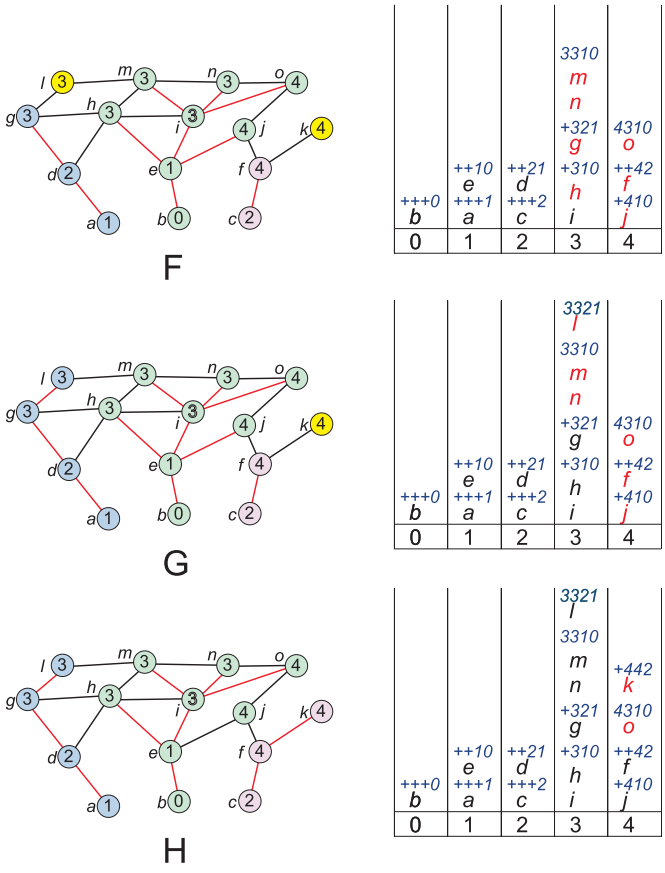

Fig. 65. Flooding from the minima of a node weighted graph controlled by a hierarchical queue. The nodes in the queue have two colors : the already expanded nodes are in dark blue and the nodes still to expand are in red. The lexicographic distance of each package of nodes is placed above the package.

Algorithm 3: The core expanding algorithm of depth 2 with tags and complete arrowing

Input: A quasi flooding graph
A hierarchical queue

Result: The labeled nodes of the catchment basins

Initialisation: Label the minima and introduce their boundary nodes into a HQ, each with a priority equal to its weight. Introduce a tag on top of the non empty hierarchical queues with the value equal to the weight of the nodes in the queue. Like that, the tag indeed indicates the lexicographic distance of the node to the regional minimum, as they belong to this minimum.

while *HQ not empty* **do**
  Create an empty "input bucket"
  Create an empty "output bucket"
  while *the next tag has not been reached in the highest priority queue* **do**
    extract the node j with the highest priority from the HQ
    put j in the "input bucket"
    for *each unlabeled neighboring node i of j* **do**
      $label(i) = label(j)$
      put i in the queue H_{ν_i} with priority ν_i
      put i in the "output bucket"
      set H_{ν_i} as active
  extract the tag in the highest priority queue; let τ be the value of this tag
  for *each active queue H_λ* **do**
    Introduce a tag equal to $\lambda \rhd \tau$ into the queue
    Set the queue as inactive
  for *each pair of neighboring nodes (i, j), j in the "input bucket" and i in the "output bucket"* **do**
    Create an arrow from i to j indicating that j is one of the nodes flooding i

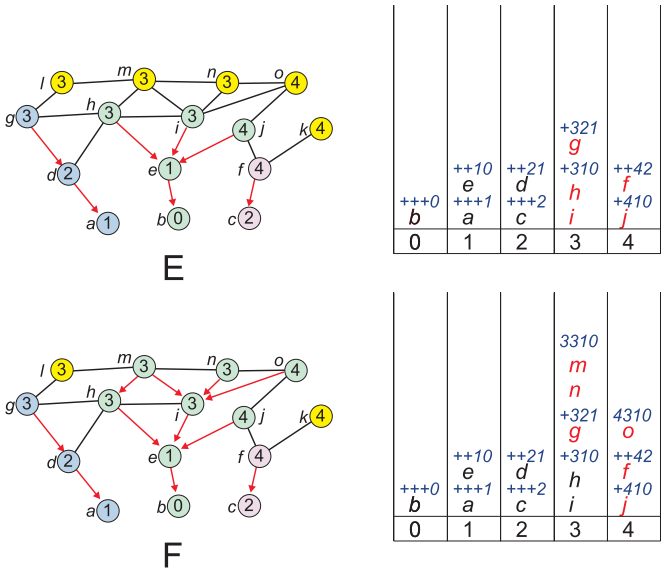

Fig. 66. Arcs are created between each flooded node and its smallest flooding nodes, in the sense of the lexicographic distance of depth ∞.

contains the nodes i and h. Expanding the node i introduces the nodes n and m. The next node to be expanded is h : having no neighboring node without label nothing happens. In our case, the input bucket contains the ndes i and h and the output bucket the nodes n and m. An arc is then created from each node in the output bucket towards each of its neighbors in the input bucket. In our cases arcs from m to h and i, and from n to i.

In practice for obtaining an optimal and simple watershed algorithm... For a node weighted graph, use the algorithm above.

For an edge weighted graph, produce the quasi-flooding graph by $\varepsilon_{ne} \downarrow G$ and apply the preceding algorithm

The hierarchical queue ordering the Moore Dijkstra algorithm The Moore-Dijkstra algorithm also may be advantageously controled by a hierarchical queue. The algorithm is classically initialized by introducing the neighboring pixels of the minima in the queue. If we take care to order the minima according the values of their nodes, and follow this order for introducing their neighboring pixels in the queue, the algorithm 4 behaves in fact as a core expanding algorithm of depth 2, that is with the help of the hierarchical queue it works as with a lexicographic distance function of depth ∞.

Algorithm 4: Catchment basins with the Moore-Dijkstra algorithm

Input: A quasi flooding graph
A hierarchical queue HQ
A domain D

Result: The labeled nodes of the catchment basins

```
1 Initialisation:
2 for each regional minimum, in the order of increasing node weights do
3 |  Label the minimum and introduce its nodes into D
4 |  Introduce each of its neighboring nodes i in the queue with priority ν_i
5 while HQ not empty do
6 |  extract the node j with the highest priority from the HQ and introduce it
  |  into D
  |  for each unlabeled neighboring node i of j do
7 |  |  label(i) = label(j)
8 |  |  put i in the queue with priority ν_i
```

"Much ado about nothing" or the silent efficiency of hierarchical queues We already suspected that in a number of publications on the watershed, the focus is put on the core of the algorithm and the role of the ordering of the nodes to be treated remains in the shadow. This disproportion is particularly spectacular for the hierarchical queues. We have just seen that by implementing the core expanding algorithm for a lexicographic distance of depth 2 (or equivalently the Voronoi tesselation based on the topographic distance) but controling its progression with a hierarchical queue, we obtain a Voronoi tesselation for a lexicographic distance of infinite depth. This means that since the paper of 1991 [36] we produced, without knowing it, the best constrained watershed possible on images and graphs (on an image, we always can do better, by enriching the neighborhood relations of each node and introducing additional directions ; in [39] we introduced chamfer neighborhoods for constructing the topographic distance).

Should we title this document "much ado about nothing", since we developed a number of techniques for constructing the best constrained watershed possible and at the same time we had for the last 20 years an optimal algorithm available ? I dont think so, as the study has given much more than just an algorithm for constructing the watershed but has given an in depth insight of the topography of node or edge weighted graphs.

27 Conclusion

Beside its theoretical interest, the paper opens also new perspectives for the watershed implementation. Local pruning operators are easily be performed by

a graphics processor. After pruning single CB may be extracted permitting local segmentations, in contrast to the flooding approach, where two competing floods have to find the divide line between them.

Acknowledgements: We thank Francis Maisonneuve for his valuable comments and suggestions for improvements of the paper.

References

1. J. Angulo and D. Jeulin. Stochastic watershed segmentation. *ISMM07 : Mathematical Morphology and its applications to Signal and Image Processing*, pages 265–276, 2007.
2. C. Berge. *Graphs*. Amsterdam: North Holland, 1985.
3. G. Bertrand. On topological watersheds. *Journal of Mathematical Imaging and Vision*, 22(3):217–230, 2005. cited By (since 1996) 9.
4. S. Beucher. *Segmentation d'Images et Morphologie Mathématique*. PhD thesis, E.N.S. des Mines de Paris, 1990.
5. S. Beucher. Watershed, hierarchical segmentation and waterfall algorithm. *ISMM94 : Mathematical Morphology and its applications to Signal Processing*, pages 69–76, 1994.
6. S. Beucher and C. Lantuéjoul. Use of watersheds in contour detection. In *Proc. Int. Workshop Image Processing, Real-Time Edge and Motion Detection/Estimation*, 1979.
7. S. Beucher and C. Lantuéjoul. Use of watersheds in contour detection. In *Watersheds of Functions and Picture Segmentation*, pages 1928–1931, Paris, May 1982.
8. S. Beucher and F. Meyer. The morphological approach to segmentation: the watershed transformation. In E. Dougherty, editor, *Mathematical morphology in image processing*, chapter 12, pages 433–481. Marcel Dekker, 1993.
9. Moga A. Bieniek, A. An efficient watershed algorithm based on connected components. *Pattern Recognition*, 33(6):907–916, 2000. cited By (since 1996) 78.
10. Gunilla Borgefors. Distance transformations in digital images. *Comput. Vision Graph. Image Process.*, 34:344–371, June 1986.
11. Otakar Boruvka. Otakar boruvka on minimum spanning tree problem: Translation of both the 1926 papers, comments, history. *DMATH: Discrete Mathematics*, 233.
12. Otakar Boruvka. O jistÃ©m problÃ©mu minimÃ¡lnÃ-m (about a certain minimal problem (in czech). *PrÃ¡ce mor. prÃ¡rodoved. spol.*, 1926.
13. Otakar Boruvka. PrÃ-spevek k reÅ¡enÃ- otÃ¡zky ekonomickÃ© stavby elektrovodnÃ-ch sÃ-tÃ- (contribution to the solution of a problem of economical construction of electrical networks)(in czech). *ElektronickÃ¡ Obzor 15*, 1926.
14. Jean Cousty, Gilles Bertrand, Laurent Najman, and Michel Couprie. Watershed cuts: Minimum spanning forests and the drop of water principle. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31:1362–1374, 2009.
15. Jean Cousty, Michel Couprie, Laurent Najman, and Gilles Bertrand. Grayscale watersheds on perfect fusion graphs. In Ralf Reulke, Ulrich Eckardt, Boris Flach, Uwe Knauer, and Konrad Polthier, editors, *Combinatorial Image Analysis*, volume 4040 of *Lecture Notes in Computer Science*, pages 60–73. Springer Berlin / Heidelberg, 2006.
16. J. Crespo, J. Serra, and R. Schafer. Image segmentation using connected filters. In *Workshop on Mathematical Morphology*, pages 52–57, Barcelona, May 1993.

17. Thomas Deschamps and Laurent D. Cohen. Fast extraction of minimal paths in 3d images and applications to virtual endoscopy. *Medical Image Analysis*, 5(4):281 – 299, 2001. <ce:title>Soft Tissue Deformation</ce:title>.
18. A.X. Falcao, J. Stolfi, and R. de Alencar Lotufo. The image foresting transform: theory, algorithms, and applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(1):19 – 29, jan 2004.
19. M. Gondran and M. Minoux. *Graphes et Algorithmes*. Eyrolles, 1995.
20. M. Grimaud. New measure of contrast : dynamics. *Image Algebra and Morphological Processing III, San Diego CA, Proc. SPIE*, 1992.
21. H. Heijmans. *Morphological Image Operators, Advances in Electronics and Electron Physics, Supplement 24, Editor-in-Chief P. Hawkes*. Boston: Academic Press, 1994.
22. H. Heijmans and C. Ronse. The algebraic basis of mathematical morphology, part I: Dilations and erosions. *Comput. Vision, Graphics and Image Processing*, 50:245–295, 1990.
23. T.C. Hu. The maximum capacity route problem. *Operation research 9*, pages 898–900, 1961.
24. Iwanowski M. Aswiercz, M. Fast, parallel watershed algorithm based on path tracing. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6375 LNCS(PART 2):317–324, 2010.
25. Piche N. Kauffmann, C. A cellular automaton for ultra-fast watershed transform on gpu. 2008.
26. J. Klein. *Conception et Réalisation d'une unité logique pour l'analyse quantitative d'images*. PhD thesis, University of Nancy, 1976.
27. C. Lantuéjoul. *La squelettisation et son application aux mesures topologiques de mosaÃŕques polycristallines*. PhD thesis, École nationale supérieure des mines de Paris, 1978.
28. C. Lantuéjoul and S. Beucher. On the use of the geodesic metric in image analysis. *J. Microsc.*, 1981.
29. F. Lemonnier. *Architecture Electronique Dédiée aux Algorithmes Rapides de Segmentation Basés sur la Morphologie Mathématique*. PhD thesis, E.N.S. des Mines de Paris, 1996.
30. F. Maisonneuve. Sur le partage des eaux. *Technical Report-Centre of Mathematical Morphology-Mines-ParisTech*, 1982.
31. Butt M.A. Maragos, P. Curve evolution, differential morphology, and distance transforms applied to multiscale and eikonal problems. *Fundamenta Informaticae*, 41(1-2):91–129, 2000. cited By (since 1996) 17.
32. P. Maragos and F. Meyer. Nonlinear PDEs and numerical algorithms for modeling levelings and reconstruction filters. In *Scale-Space Theories in Computer Vision*, Lecture Notes in Computer Science 1682, pages 363–374. Springer, 1999.
33. B. Marcotegui and S. Beucher. Fast implementation of waterfalls based on graphs. *ISMM05 : Mathematical Morphology and its applications to Signal Processing*, pages 177–186, 2005.
34. F. Meyer. Skeletons and perceptual graphs. *Signal Processing*, 16(4):690–710, April 1989.
35. F. Meyer and S. Beucher. Morphological segmentation. *JVCIP*, 1(1):21–46, Sept. 1990.
36. F. Meyer. Un algorithme optimal de ligne de partage des eaux. In *Proceedings* $8^{\underline{ème}}$ *Congrès AFCET, Lyon-Villeurbanne*, pages 847–857, 1991.
37. F. Meyer. Color image segmentation. In *IEE Fourth International Conference on Image Processing and its Applications*, pages 303–306, April 1992.

38. F. Meyer. Minimal spanning forests for morphological segmentation. *ISMM94 : Mathematical Morphology and its applications to Signal Processing*, pages 77–84, 1994.
39. F. Meyer. Topographic distance and watershed lines. *Signal Processing*, pages 113–125, 1994.
40. F. Meyer. The levelings. In H. Heijmans and J. Roerdink, editors, *Mathematical Morphology and Its Applications to Image Processing*, pages 199–207. Kluwer, 1998.
41. Fernand Meyer. Grey-weighted, ultrametric and lexicographic distances. In Christian Ronse, Laurent Najman, and Etienne DecenciÃ¨re, editors, *Mathematical Morphology: 40 Years On*, volume 30 of *Computational Imaging and Vision*, pages 289–298. Springer Netherlands, 2005.
42. Fernand Meyer. Flooding and segmentation. In John Goutsias, Luc Vincent, and Dan S. Bloomberg, editors, *Mathematical Morphology and its Applications to Image and Signal Processing*, volume 18 of *Computational Imaging and Vision*, pages 189–198. 2002.
43. Fernand Meyer. The watershed concept and its use in segmentation : a brief history. *(arXiv:1202.0216v1)*, 2012.
44. E.F. Moore. The shortest path through a maze. In *Proc. Int. Symposium on Theory of Switching*, volume 30, pages 285–292, 1957.
45. L. Najman. *Morphologie Mathématique: de la segmentation d'images à l'analyse multivoque*. PhD thesis, Université Paris-Dauphine, 1994.
46. L. Najman. Geodesic saliency of watershed edges and hierarchical segmentation. *IEEE Trans. Pattern Anal. Machine Intell*, 16(3):175–182, 1996.
47. Laurent Najman and Michel Schmitt. Watershed of a continuous function. *Signal Processing*, 38(1):99 – 112, 1994. Mathematical Morphology and its Applications to Signal Processing.
48. Hieu Tat Nguyen, Marcel Worring, and Rein van den Boomgaard. Watersnakes: Energy-driven watershed segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:330–342, 2003.
49. D. Noguet. A massively parallel implementation of the watershed based on cellular automata. In *Application-Specific Systems, Architectures and Processors, 1997. Proceedings., IEEE International Conference on*, pages 42 –52, jul 1997.
50. Jos B. T. M. Roerdink and Arnold Meijster. The watershed transform: Definitions, algorithms and parallelization strategies. *Fundamenta Informaticae*, 41:187–228, 2001.
51. P. Salembier and J. Serra. Flat zones filtering, connected operators and filters by reconstruction. *IEEE Transactions on Image Processing*, 3(8):1153–1160, August 1995.
52. J. Serra, editor. *Image Analysis and Mathematical Morphology. II: Theoretical Advances*. Academic Press, London, 1988.
53. C. Vachier. *Extraction de Caractéristiques, Segmentation d'Image et Morphologie Mathématique*. PhD thesis, E.N.S. des Mines de Paris, 1995.
54. C. Vachier and F. Meyer. Extinction values: A new measurement of persistence. In I. Pitas, editor, *1995 IEEE Workshop on Nonlinera Signal and Image Processing*, pages 254–257, 1995.
55. Corinne Vachier and Fernand Meyer. The viscous watershed transform. *Journal of Mathematical Imaging and Vision*, 22:251–267, 2005.
56. Jeulin D. Vincent, L. Minimal paths and crack propagation simulations. *Acta Stereologica*, 8(2 II):487–494, 1989.
57. L. Vincent. *Algorithmes Morphologiques à Base de Files d'Attente et de Lacets. Extension aux Graphes*. PhD thesis, E.N.S. des Mines de Paris, 1990.

58. L. Vincent. Morphological area openings and closings for grayscale images. *Shape in Picture, NATO Workshop, Driebergen*, Sept. 1992.
59. Soille Pierre Vincent, Luc. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991.
60. Luc Vincent. Minimal path algorithms for the robust detection of linear features in gray images. In *Proceedings of the fourth international symposium on Mathematical morphology and its applications to image and signal processing*, ISMM '98, pages 331–338, Norwell, MA, USA, 1998. Kluwer Academic Publishers.
61. F. Zanoguera, B. Marcotegui, and F. Meyer. A segmentation pyramid for the interactive segmentation of 3-d images and video sequences. In John Goutsias, Luc Vincent, and Dan S. Bloomberg, editors, *Mathematical Morphology and its Applications to Image and Signal Processing*, volume 18 of *Computational Imaging and Vision*, pages 223–232. Springer US, 2002.
62. L. Najman and h. Talbot (ed) Mathematical morphology *Wiley editor*, 2012