Efficient algorithms for hierarchical graph-based segmentation relying on the Felzenszwalb-Huttenlocher dissimilarity

Edward Jorge Yuri Cayllahua Cahuina, Jean Cousty, Yukiko Kenmochi, Arnaldo Albuquerque de Araújo, Guillermo Cámara-Chávez, Silvio Jamil F. Guimarães

International Journal of Pattern Recognition and Artificial Intelligence

Efficient algorithms for hierarchical graph-based segmentation relying on the Felzenszwalb-Huttenlocher dissimilarity

Edward Cayllahua Cahuina

Universidade Federal de Minas Gerais, Computer Science Department, 31270-901 Belo Horizonte, Brazil and Université Paris-Est, ESIEE Paris, F-93162 Noisy-le-Grand, France

Jean Cousty

Université Paris-Est, LIGM (UMR 8049), CNRS, ENPC, ESIEE Paris, UPEM, F-93162, Noisy-le-Grand, France

Université Paris Descartes, Laboratoire MAP5 (UMR 8145), 12 Rue de l'École de Médecine, 75006 Paris , France

Yukiko Kenmochi

Université Paris-Est, LIGM (UMR 8049), CNRS, ENPC, ESIEE Paris, UPEM, F-93162, Noisy-le-Grand, France

Arnaldo de Albuquerque Araújo

Universidade Federal de Minas Gerais, Computer Science Department, 31270-901 Belo Horizonte, Brazil and Université Paris-Est, ESIEE Paris, F-93162 Noisy-le-Grand, France

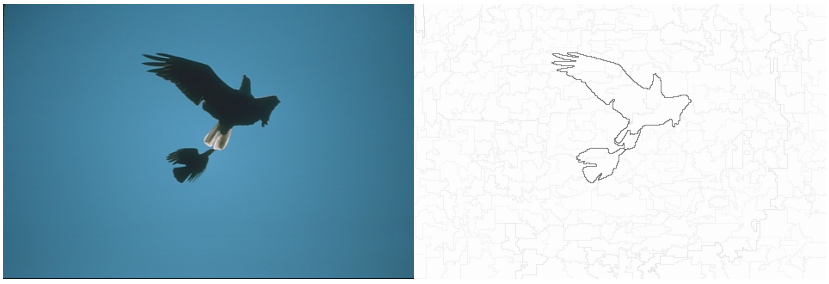
Guillermo Cámara-Chávez

Universidade Federal de Ouro Preto, Computer Science Department, 35400-000 Ouro Preto, Brazil

Silvio Jamil F. Guimarães

PUC Minas - ICEI - Computer Science Department - VIPLAB, 30535-065 Belo Horizonte, Brazil and Université Paris-Est, ESIEE Paris, F-93162 Noisy-le-Grand, France

Hierarchical image segmentation provides a region-oriented scale-space, *i.e.*, a set of image segmentations at different detail levels in which the segmentations at finer levels are nested with respect to those at coarser levels. However, most image segmentation algorithms, among which a graph-based image segmentation method relying on a region merging criterion was proposed by Felzenszwalb-Huttenlocher in 2004, do not lead to a hierarchy. In order to cope with a demand for hierarchical segmentation, Guimarães *et al.* proposed in 2012 a method for hierarchizing the popular Felzenszwalb-Huttenlocher method, without providing an algorithm to compute the proposed hierarchy. This article is devoted to provide a series of algorithms to compute the result of this hierarchical graph-based image segmentation method efficiently, based mainly on two ideas: optimal dissimilarity measuring and incremental update of the hierarchical structure. Experiments show that, for an image of size 321 × 481 pixels, the most efficient algorithm produces the result in half a second whereas the most naive one requires more than four



(a) Original image

(b) Hierarchical segmentation depicted as a saliency map

(c) Some levels of this hierarchical segmentation

Fig. 1: Illustration of a hierarchical image segmentation.

hours.

Keywords: Image segmentation; hierarchical analysis; quasi-flat zone; incremental algorithm.

1. Introduction

A hierarchical image segmentation is a series of image segmentations at different detail levels where the segmentations at higher detail levels are produced by merging regions from segmentations at finer detail levels. Consequently, the regions at finer detail levels are nested in regions at coarser levels. The level of a segmentation in a hierarchy is also called an *observation scale*. An example of hierarchical image segmentation is illustrated in Fig. 1.

Hierarchical image segmentation provides a multi-scale approach to image analysis. Hierarchical image analysis was pioneered by Ref. 23 and has received a lot of attention since then, as attested by the popularity of Ref. 1. In Ref. 39, the global

information is used to create the initial regions and then the region merging process is treated as a series of optimization problems. Mathematical morphology is also used in hierarchical image analysis with, *e.g.*, hierarchical watersheds in Refs. 2, 21, 18, 5, 25, binary partition trees in Refs. 36, 35, regular and irregular pyramids in Ref. 17, scale-set theory in Ref. 10, quasi-flat zones hierarchies in Ref. 19, tree-based shape spaces in Ref. 41. Other methods for hierarchical image segmentation consider multiscale combinatorial grouping and region merging procedures in Ref. 29 and in Refs. 34, 26, 43, respectively. Hierarchical image analysis has been used in computer vision to solve practical problems such as occlusion boundary detection in Ref. 16, image simplification in Refs. 10, 26, 38, object proposal in Ref. 29, visual saliency estimation in Ref. 42, attribute profile for image classification in Ref. 28.

According to Guigues *et al.*, a hierarchy should satisfy two important principles. First, the *causality principle* which states that a contour present at an observation scale k_1 should also be present at any scale k_2, such that $k_2 < k_1$. Second, the *location principle* which states that contours should neither move nor deform from one observation scale to another.[10] Any hierarchy should comply with these principles for multi-scale analysis.

In Ref. 11 (see Ref. 13 for its preliminary version), the quasi-flat zone hierarchy is used to perform a hierarchical image segmentation. This work relies on the graph-based (GB) image segmentation algorithm proposed in Ref. 8. The GB algorithm uses a merging predicate to decide if, at a certain scale parameter, two adjacent regions of an image should be merged into a single one, and thus produces a segmented image. In its original form, the algorithm does not directly lead to a hierarchical image segmentation. This is confirmed in Ref. 11, the original GB algorithm does not comply with the causality and location principles;[8] first, when increasing the scale parameter it produces a larger number of regions and thus violates the causality principle; second, it was also observed that the contours may deform as the scale parameter varies and thus violate the location principle. This motivated Guimarães *et al.* to formulate a dissimilarity measure based on the GB region merging predicate.[11] Based on this dissimilarity criterion and on the notion of a quasi-flat zones hierarchy, Guimarães et al. have provided a hierarchical graph-based image segmentation (HGB) method in Ref. 11.

Assessing the quality of the results of hierarchical segmentation methods is a difficult task (see Refs. 33, 25, 27, 31, 30, 32) and it is assessed in Ref. 11 that the HGB method produces satisfactory results. Nonetheless, a practical algorithm to compute efficiently results of the HGB method is not provided in Ref. 11. The core of the HGB method is based on solving a minimization problem whose solution is the minimum observation scale at which adjacent regions in the image have to be merged. To solve this minimization, the naive method consists of considering all positive real values to find such minimum observation scale.

In this article, we study the HGB method focusing on two problems that make difficult its implementation. A first difficulty is related to solving the minimization problem involved in the HGB method for which no efficient algorithmic solution

is given in Ref. 11. We analyze this minimization process and propose three algorithms that solve it. The first one solves the minimization by searching the result in a sufficiently large space of possible values. We then reduce this search space to avoid redundant computations, leading to two efficient algorithms. The second problem is related to the quasi-flat zone computation. One approach can be to use an efficient algorithm, as the one proposed in Ref. 20, to compute it at every iteration of the HGB method. However, efficiency can be improved by only updating at each iteration the existing quasi-flat zone hierarchy instead of recomputing it from scratch. This is done with a procedure similar to the one proposed in Refs. 15, 40. Overall, the most efficient proposed algorithm computes the result of HGB method for an image of size 321×481 pixels in about half a second whereas it takes over four hours with the most naive algorithm.

This article is an extension of the conference article presented in Ref. 4 that proposes a series of algorithms to compute the HGB method. The new contributions of this article are: introducing a general framework for solving the minimization involved in the HGB method, giving the proofs of all the properties that were used in Ref. 4, analysing the algorithm complexity for each of the algorithms and showing more experimental results concerning the execution time on a full dataset.

The remainder of this article is organized as follows. Section 2 gives us the basic notions for hierarchical graph-based image segmentation and, based on these notions, also introduces the HGB method. Section 3 discusses the implementation of the HGB method and presents the algorithms to solve the minimization problem involved in the HGB method and the quasi-flat zone computation. Section 4 assesses the efficiency comparison of all the algorithms presented in Section 3 measuring their execution times. Section 5 concludes the article and also gives ideas about future research directions.

2. Hierarchical graph-based image segmentation

This section aims at explaining the method of hierarchical graph-based image segmentation (HGB) presented in Ref. 13. The hierarchy is constructed from an image via a graph representation, based on the notion of a quasi-flat zone hierarchy.[19] We first give a series of necessary notions, quasi-flat zones hierarchies, and then explain the HGB method.

2.1. *Basic notions*

2.1.1. *Hierarchies*

Given a finite set V, a *partition* of V is a set $\mathbf{P}$ of nonempty disjoint subsets of V whose union is V. Any element of $\mathbf{P}$ is called a *region* of $\mathbf{P}$. Given two partitions $\mathbf{P}$ and $\mathbf{P}'$ of V, $\mathbf{P}'$ is said to be a refinement of $\mathbf{P}$, denoted by $\mathbf{P}' \preceq \mathbf{P}$, if any region of $\mathbf{P}'$ is included in a region of $\mathbf{P}$. A hierarchy on V is a sequence $\mathcal{H} = (\mathbf{P}_0, \ldots, \mathbf{P}_\ell)$ of partitions of V, such that $\mathbf{P}_{i-1} \preceq \mathbf{P}_i$, for any $i \in \{1, \ldots, \ell\}$.

2.1.2. *Graph and connected-component partition*

A *graph* is a pair $G = (V, E)$ where V is a finite set and E is a subset of $\{\{x, y\} \subseteq V | x \neq y\}$. Each element of V is called a *vertex* of G, and each element of E is called an *edge* of G. A subgraph of G is a graph (V', E') such that $V' \subseteq V$ and $E' \subseteq E$. If X is a graph, its vertex and edge sets are denoted by $V(X)$ and $E(X)$, respectively.

Let x and y be two vertices of a graph G. A *path from x to y* in G is a sequence $(x_0, \ldots, x_m)$ of vertices of G such that $x_0 = x$, $x_m = y$ and $\{x_{i-1}, x_i\}$ is an edge of G for any i in $\{1, \ldots, m\}$. The graph G is *connected* if, for any vertices x and y of G, there exists a path from x to y. Let A be a subset of $V(G)$. The graph induced by A in G is the graph whose vertex set is A and whose edge set contains any edge of G made of two elements in A. If the graph induced by A is connected, then we say that A is *connected*. The subset A of $V(G)$ is a *connected component* of G if it is connected for G and maximal for this property. Y of $V(G)$, if Y is a connected superset of X, then $Y = X$. We denote by $\mathbf{C}(G)$ the set of all connected components of G. Note that $\mathbf{C}(G)$ is a partition of $V(G)$, which is called the *connected-component partition induced by G*.

2.1.3. *Quasi-flat zone hierarchies*

Given a graph $G = (V, E)$, let w be a map from E into the set $\mathbb{R}$ of real numbers. For any edge u of G, the value $w(u)$ is called the weight of u (for w), and the pair (G, w) is called an *edge-weighted graph*. We now explain how to make from an edge-weighted graph a series of connected-component partitions, which constitutes a hierarchy. Such a hierarchy is called a quasi-flat zone hierarchy of (G, w) and the quasi-flat zone hierachy transform is a bijection beetween the hierarchies and a subset of the edge weighted graphs (called the saliency maps). Hence, any edge-weighted graph induces a quasi-flat zone hierarchy and any hierarchy $\mathcal{H}$ can be represented by an edge-weighted graph whose quasi-flat zone hierarchy is precisely $\mathcal{H}$ (see Ref. 6 for more details). This bijection indeed allows us to handle quasi-zone hierarchies through edge-weighted graphs. In other words, when an edge-weighted graph is created from an input image, the bijection signifies that this initial graph already possesses a quasi-flat zone hierarchy and that the procedure of hierarchical image segmentation can be interpreted as a modification of the initial hierarchical structure by changing edge weights on the graph.

Given an edge-weighted graph (G, w), let X be a subgraph of G and let λ be a value of $\mathbb{R}$. The *λ-level edge set* of X for w is defined by

$$w_\lambda(X) = \{u \in E(X) \mid w(u) < \lambda\}, \tag{1}$$

and the *λ-level graph* of X for w is defined as the subgraph $w^V_\lambda(X)$ of X, such that

$$w^V_\lambda(X) = (V(X), w_\lambda(X)). \tag{2}$$

Then, the connected-component partition $\mathbf{C}(w_\lambda^V(X))$ induced by $w_\lambda^V(X)$ is called the *λ-level partition* of X for w.

As we consider only finite graphs and hierarchies, the set of considered level values is reduced to a finite subset of $\mathbb{R}$ that is denoted by $\mathbb{E}$ in the remaining parts of this article. In order to browse the values of this set and to round real values to values of $\mathbb{E}$, we define, for any $\lambda \in \mathbb{R}$:

$$\begin{aligned} \mathrm{p}_{\mathbb{E}}(\lambda) &= \max\{\mu \in \mathbb{E} \cup \{-\infty\} \mid \mu < \lambda\}; \\ \mathrm{n}_{\mathbb{E}}(\lambda) &= \min\{\mu \in \mathbb{E} \cup \{\infty\} \mid \mu > \lambda\}; \text{ and} \\ \hat{\mathrm{n}}_{\mathbb{E}}(\lambda) &= \min\{\mu \in \mathbb{E} \cup \{\infty\} \mid \mu \geq \lambda\}. \end{aligned}$$

Let (G, w) be an edge-weighted graph and let X be a subgraph of G. The sequence, denoted by $\mathcal{QFZ}(X, w)$, of all λ-level partitions of X for w ordered by increasing value of λ, namely

$$\mathcal{QFZ}(X, w) = (\mathbf{C}(w_\lambda^V(X)) \mid \lambda \in \mathbb{E} \cup \{\infty\}), \tag{3}$$

is a hierarchy, called the *quasi-flat zone hierarchy of X for w*. Let $\mathcal{H}$ be the quasi-flat zone hierarchy of G for w. Given a vertex x of G and a value λ in $\mathbb{E}$, the region that contains x in the λ-level partition of the graph G is denoted by $\mathcal{H}_x^\lambda$.

Important notations and remarks. In the remaining parts of this article, the symbol G denotes a connected graph, the symbol w denotes a map from E into $\mathbb{R}$, and the symbol T denotes a minimum spanning tree of (G, w). It has been shown in Ref. 6 that the quasi-flat zone hierarchy $\mathcal{QFZ}(T, w)$ of T for w is the same as the quasi-flat zone hierarchy $\mathcal{QFZ}(G, w)$ of G for w. This indicates that the quasi-flat zone hierarchy for G can be handled by its minimum spanning tree.

2.2. *Hierarchical graph-based segmentation method*

In this article, we consider that the input is the edge-weighted graph (G, w) representing an image, where the pixels correspond to the vertices of G and the edges link adjacent pixels. The weight of each edge is given by a dissimilarity measure between the linked pixels such as the absolute difference of intensity between them.

Before explaining HGB method, we first describe the following observation scale dissimilarity defined in Ref. 11, which is required by the method and whose idea originates from the region merging criterion proposed in Ref. 8.

2.2.1. *Observation scale dissimilarity*

As mentioned above, the graph-based hierarchical image segmentation method is based on the idea of renewing graph-edge weights, and this renewal is made depending on the following region dissimilarity. Indeed, with the Felzenszwalb-Huttenlocher image segmentation algorithm defined in Ref. 8, two regions of an image are merged based on a region merging predicate. This predicate was later reformulated as an observation scale dissimilarity measure to produce the hierarchical segmentation of an image as follows.

Method 1: HGB method

Input : A minimum spanning tree T of an edge-weighted graph (G, w)
Output: A hierarchy $\mathcal{H} = \mathcal{QFZ}(T, f)$

1 **for** *each* $u \in E(T)$ **do** $f(u) := \max\{\lambda \in \mathbb{E}\}$;
2 **for** *each* $u \in E(T)$ *in non-decreasing order for* w **do**
3 $\quad \mathcal{H} := \mathcal{QFZ}(T, f)$;
4 $\quad f(u) := \mathrm{p}_{\mathbb{E}}(\lambda^{\star}_{\mathcal{H}}(u))$;
5 **end**
6 $\mathcal{H} := \mathcal{QFZ}(T, f)$;

Let R_1 and R_2 be two adjacent regions, the dissimilarity measure compares the so-called inter-component and within-component differences.[8] The *inter-component difference* between R_1 and R_2 is defined by

$$\Delta_{inter}(R_1, R_2) = \min\{w(\{x, y\}) \,|x \in R_1, y \in R_2, \{x, y\} \in E(T)\},$$

while the *within-component difference* of a region R is defined by

$$\Delta_{intra}(R) = \max\{w(\{x, y\}) \,|x, y \in R, \{x, y\} \in E(T)\}.$$

It leads to the *observation scale of* R_1 *relative to* R_2, defined by

$$S_{R_2}(R_1) = (\Delta_{inter}(R_1, R_2) - \Delta_{intra}(R_1))\,|R_1|,$$

where $|R_1|$ is the cardinality of R_1. Then, a symmetric metric between R_1 and R_2, called the *observation scale dissimilarity between* R_1 *and* R_2, is defined by

$$D(R_1, R_2) = \max\{S_{R_2}(R_1), S_{R_1}(R_2)\}. \tag{4}$$

This dissimilarity is used to determine if two regions should be merged or not at a certain observation scale in the following.

2.2.2. *HGB Method*

The HGB method is presented in Method 1. The input is an image represented by a graph G with its associated weight function w, where the minimum spanning tree T of G is taken indeed. From (T, w), HGB method computes a new weight function f which leads to a new hierarchy $\mathcal{H} = \mathcal{QFZ}(T, f)$. The resulting hierarchy $\mathcal{H}$ is considered as the hierarchical image segmentations of the initial image. Thus, the core of the method is the generation of the weight function f for T.

To compute the new map f, the HGB method first initializes all values of f to infinity (see Line 1). Then, an observation scale value $f(u)$ is computed for each edge $u \in E(T)$ in non-decreasing order with respect to the original weight w (see Line 2). Note that each iteration in the loop requires computing the hierarchy

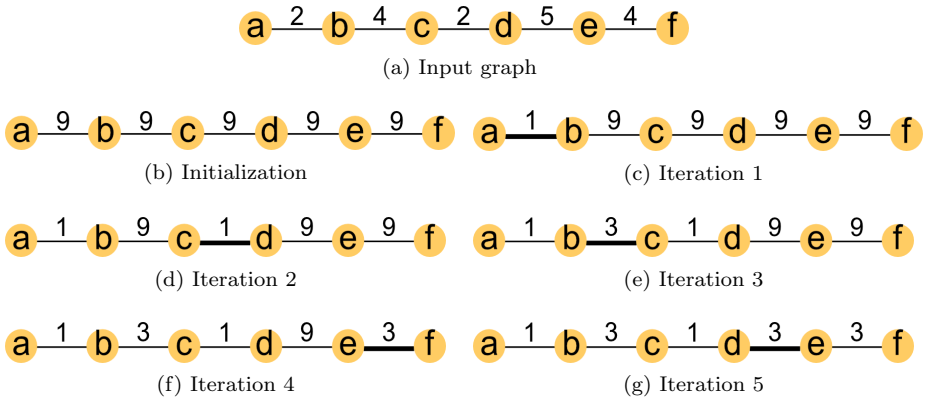


Fig. 2: Illustration of Method 1 with $\mathbb{E} = \{0, 1, \ldots, 9\}$: (a) the input graph (T, w), (b-g) the graph (T, f) at each iteration of Method 1 and (g) the resulting quasi-flat zone hierarchy corresponding to graph (T, f).

$\mathcal{H} = \mathcal{QFZ}(T, f)$ (see Line 3). Once $\mathcal{H}$ is obtained, the value $\lambda^\star_{\mathcal{H}}(u)$ of a finite subset $\mathbb{E}$ of $\mathbb{R}$ is obtained by the minimization:

$$\lambda^\star_{\mathcal{H}}(\{x, y\}) = \min\left\{\lambda \in \mathbb{E} \mid D\left(\mathcal{H}^\lambda_x, \mathcal{H}^\lambda_y\right) \leq \lambda\right\}. \tag{5}$$

We first consider the regions $\mathcal{H}^\lambda_x$ and $\mathcal{H}^\lambda_y$ at a level λ. Using the dissimilarity measure D, we check if $D\left(\mathcal{H}^\lambda_x, \mathcal{H}^\lambda_y\right) \leq \lambda$. Equation (5) states that $\lambda^\star_{\mathcal{H}}(\{x, y\})$ is the minimum value λ for which this assertion holds true. Observe that the minimization involved in Equation (5) has a solution only if the maximum of $\mathbb{E}$ is greater than the maximum possible dissimilarity value. In the following, we assume that this assumption always holds true. Fig. 2 illustrates an example of application of Method 1.

It should be probably mentioned that the non-hierarchical method proposed by Felzenswalb *et al.* guarantees to obtain partitions that are neither too fine nor too coarse for a given scale.[8] On the other hand, the hierarchical method cannot provide simultaneously both of similar properties that are extended to hierarchies.[11] However, the later method allows us to find maximal not-too-coarse hierarchies (see Ref. 11 for more details).

As mentioned above, Guimarães *et al.* did not provide a practically efficient algorithm to compute Method 1. In order to fill this gap, the problem is twofold. Indeed, it is necessary to propose efficient (i.e., exact and fast) algorithms for (i) solving the minimization involved in Equation (5); and (ii) computing the quasi-flat zone hierarchy $\mathcal{QFZ}(T, f)$ at each iteration of Method 1 (Lines 3 and 6).

3. Algorithms for HGB method

In this section, we investigate algorithms to compute the results of HGB method. After giving a naive algorithm in Section 3.1, we first introduce the common notions

for solving efficiently the minimization involved at Line 4 of Method 1 in Section 3.2, and we present two different ideas to improve the naive algorithm in Sections 3.3 and 3.4 using the notions of Section 3.2. In Section 3.5, we present non-incremental and incremental algorithms to obtain the quasi-flat zone hierarchy of a weight map as requested at Lines 3 and 6 of Method 1.

3.1. *Naive minimization algorithm*

We first present a naive algorithm, namely Algorithm 1, to compute the value $\lambda^\star_{\mathcal{H}}(\{x, y\})$ given a hierarchy $\mathcal{H}$ and an edge $\{x, y\}$. According to Equation (5), it simply consists of considering the values of $\mathbb{E}$ in increasing order until finding a value $\lambda \in \mathbb{E}$ such that $D\left(\mathcal{H}^\lambda_x, \mathcal{H}^\lambda_y\right) \leq \lambda$. We remark that, when $\mathbb{E}$ is a set of consecutive integers, for any λ in $\mathbb{E}$, the result of $\mathrm{n}_{\mathbb{E}}(\lambda)$ and $\mathrm{p}_{\mathbb{E}}(\lambda)$ can be obtained with the simple integer instruction $\lambda + 1$ and $\lambda - 1$, respectively.

As said in Ref. 3, the size and the maximal edge-value of each region of $\mathcal{H}$ can be computed on the fly at Line 3 of Method 1. Thus, once the regions $\mathcal{H}^\lambda_x$ and $\mathcal{H}^\lambda_y$ are identified the dissimilarity between them can be computed in constant time. Furthermore, if the hierarchy $\mathcal{H}$ is represented by its dendrogram, the regions $\mathcal{H}^\lambda_x$ and $\mathcal{H}^\lambda_y$ can be obtained at each iteration of the main loop of *Algorithm* 1 in constant time (more details about the implementation of such operation are provided in the following Section 3.4). Therefore, the time complexity of Algorithm 1 is $O(|\mathbb{E}|)$.

Algorithm 1: HGB Naive minimization of Equation 5

Input : A hierarchy $\mathcal{H}$, an edge $\{x, y\}$
Output: The value $\lambda^\star$ such that $\lambda^\star = \lambda^\star_{\mathcal{H}}(\{x, y\})$

```
1 λ* := min{λ ∈ E} ;
2 while D(H^λ_x, H^λ_y) > λ* do
3 |   λ* := n_E(λ*) ;
4 end
```

3.2. *Stable intervals and stable partitions*

In this section, we propose a general framework to solve efficiently the minimization problem presented in Equation (5). To this end, we establish (Property 3) that on certain subdomains, called stable intervals, the solution depends only on the bounds of the subdomain instead of all of its elements, reducing the computation of the solution in such subdomain to a constant number of operations instead of a number of operations which depends linearly on the size of the subdomain. Then, we state (Theorem 5) that the problem on the whole domain can be solved by considering any partition of the domain into stable intervals and by solving the problem in each of these stable intervals. This theorem is the fundamental result which allows us for

proposing efficient minimization algorithms in Sections 3.3 and 3.4. More precisely, in these following sections, partitions of the domain in stable intervals that can be handled efficiently in a computerized procedure are presented.

Let λ_1 and λ_2 be any two real numbers in $\mathbb{E} \cup \{-\infty\}$ such that $\lambda_1 < \lambda_2$. We denote by $]\!]\lambda_1, \lambda_2]\!]_{\mathbb{E}}$ the subset of $\mathbb{E}$ that contains every element of $\mathbb{E}$ that is both greater than λ_1 and not greater than λ_2:

$$]\!]\lambda_1, \lambda_2]\!]_{\mathbb{E}} = \{\lambda \in \mathbb{E} \mid \lambda_1 < \lambda \leq \lambda_2\}. \tag{6}$$

We say that a subset I of $\mathbb{E}$ is an *open-closed interval of* $\mathbb{E}$, or simply an *interval*, if there exists two real values λ_1 and λ_2 such that I is equal to $]\!]\lambda_1, \lambda_2]\!]_{\mathbb{E}}$.

Definition 1 (stable interval). Let f be any map from $E(T)$ in $\mathbb{E}$, let $u = \{x, y\}$ be any edge in $E(T)$, and let $I =]\!]\lambda_1, \lambda_2]\!]_{\mathbb{E}}$ be any interval. We say that I is a *stable interval for* (f, u) if, for any two values λ and λ' in I, the two following statements hold true:

(1) the regions containing x in the λ-level partition of (T, f) and in the λ'-level partition of (T, f) are equal; and
(2) the regions containing y in the λ-level partition of (T, f) and in the λ'-level partition of (T, f) are equal.

In other words, the interval $I =]\!]\lambda_1, \lambda_2]\!]_{\mathbb{E}}$ is a stable interval for (f, u) if, for any two values λ and λ' in I, we have $\mathcal{H}_x^{\lambda} = \mathcal{H}_x^{\lambda'}$ and $\mathcal{H}_y^{\lambda} = \mathcal{H}_y^{\lambda'}$, where $\mathcal{H}$ is the quasi-flat zone hierarchy of T for f. Hence, the following lemma can be straightforwardly deduced from the definition of the dissimilarity measure D.

Lemma 2. *Let f be any map from $E(T)$ in $\mathbb{E}$ and let $\mathcal{H}$ be the quasi-flat zone hierarchy of T for f, let $u = \{x, y\}$ be any edge in $E(T)$, and let $I =]\!]\lambda_1, \lambda_2]\!]_{\mathbb{E}}$ be a stable interval for (f, u). Then, for any λ in I, we have: $D(\mathcal{H}_x^{\lambda}, \mathcal{H}_y^{\lambda}) = D(\mathcal{H}_x^{\lambda_2}, \mathcal{H}_y^{\lambda_2})$.*

The following property firstly shows a necessary and sufficient condition on which the minimization problem presented in Equation (5) admits a solution when its domain of definition is restricted to a stable interval. Moreover, when the problem admits a solution over a certain stable interval, the following property provides an expression of this solution that depends only on the bounds of the considered stable interval.

Property 3. Let f be any map from $E(T)$ in $\mathbb{E}$, let $\mathcal{H}$ be the quasi-flat zone hierarchy of T for f, let $u = \{x, y\}$ be any edge in $E(T)$, and let $I =]\!]\lambda_1, \lambda_2]\!]_{\mathbb{E}}$ be a stable interval for (f, u). Then, the two following statements hold true:

(1) the set $\{\lambda \in I \mid D(\mathcal{H}_x^{\lambda}, \mathcal{H}_y^{\lambda}) \leq \lambda\}$ is nonempty if and only if $D(\mathcal{H}_x^{\lambda_2}, \mathcal{H}_y^{\lambda_2})$ is not greater than λ_2 (*i.e.*, $D(\mathcal{H}_x^{\lambda_2}, \mathcal{H}_y^{\lambda_2}) \leq \lambda_2$); and
(2) if $D(\mathcal{H}_x^{\lambda_2}, \mathcal{H}_y^{\lambda_2}) \leq \lambda_2$, then

$$\min\left\{\lambda \in I \mid D(\mathcal{H}_x^{\lambda}, \mathcal{H}_y^{\lambda}) \leq \lambda\right\} = \max\left\{\mathrm{n}_{\mathbb{E}}\left(\lambda_1\right), \hat{\mathrm{n}}_{\mathbb{E}}\left(D(\mathcal{H}_x^{\lambda_2}, \mathcal{H}_y^{\lambda_2})\right)\right\}. \tag{7}$$

Proof. In order to prove Property 3, we consider three distinct cases. For each of these cases, we prove that $\{\lambda \in I \mid D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda) \leq \lambda\}$ is either empty or nonempty and in the cases where it is nonempty, we establish that Equation (7) holds true. For each of these three cases, the associated proofs are graphically illustrated in Fig. 3.

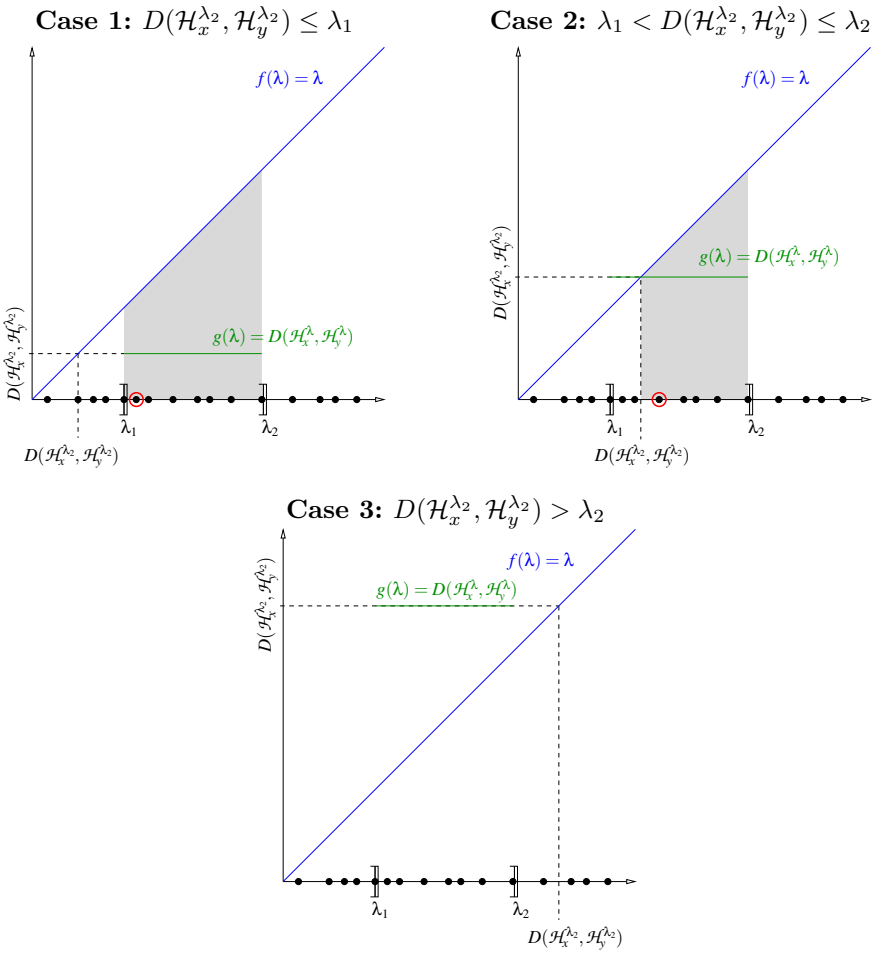


Fig. 3: Graphical illustrations for the proof of Property 3. The elements of $\mathbb{E}$ are represented by dots on the horizontal axis of real values. The subdomain of E which contains every element λ of $I =]\!]\lambda_1, \lambda_2]\!]_{\mathbb{E}}$ such that $D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda) \leq \lambda$ is represented by the gray zones and, when this subdomain is not empty, the unique solution to $\min\{\lambda \in I \mid D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda) \leq \lambda\}$ is indicated by the circled dot.

(1) Let us first assume that $D(\mathcal{H}_x^{\lambda_2}, \mathcal{H}_y^{\lambda_2}) \leq \lambda_1$. By Lemma 2, we deduce that, for

any $\lambda \in]\!]\lambda_1, \lambda_2]\!]_{\mathbb{E}}$, we have $D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda) \leq \lambda_1$. Hence, for any λ in $I =]\!]\lambda_1, \lambda_2]\!]_{\mathbb{E}}$, we have $D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda) \leq \lambda$. Since λ_2 belongs to I, in this case, the set $\{\lambda \in I \mid D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda) \leq \lambda\}$ is nonempty and we deduce that $\min\{\lambda \in I \mid D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda) \leq \lambda\}$ is the minimum of I, which is precisely $\mathrm{n}_{\mathbb{E}}(\lambda_1)$. Thus, we have $\min\{\lambda \in I \mid D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda) \leq \lambda\} = \mathrm{n}_{\mathbb{E}}(\lambda_1)$. Furthermore, since $D(\mathcal{H}_x^{\lambda_2}, \mathcal{H}_y^{\lambda_2}) \leq \lambda_1$, we have $\hat{\mathrm{n}}_{\mathbb{E}}\left(D(\mathcal{H}_x^{\lambda_2}, \mathcal{H}_y^{\lambda_2})\right) \leq \hat{\mathrm{n}}_{\mathbb{E}}(\lambda_1)$. Therefore, since λ belongs to $\mathbb{E}$, we deduce that $\hat{\mathrm{n}}_{\mathbb{E}}\left(D(\mathcal{H}_x^{\lambda_2}, \mathcal{H}_y^{\lambda_2})\right) < \mathrm{n}_{\mathbb{E}}(\lambda_1)$. Hence, in the considered case, we have $\min\left\{\lambda \in I \mid D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda) \leq \lambda\right\} = \max\left\{\mathrm{n}_{\mathbb{E}}(\lambda_1), \hat{\mathrm{n}}_{\mathbb{E}}\left(D(\mathcal{H}_x^{\lambda_2}, \mathcal{H}_y^{\lambda_2})\right)\right\}$.

(2) Let us now assume that $\lambda_1 < D(\mathcal{H}_x^{\lambda_2}, \mathcal{H}_y^{\lambda_2}) \leq \lambda_2$. Since λ_2 belongs to $I =]\!]\lambda_1, \lambda_2]\!]_{\mathbb{E}}$, we deduce that, in this case, the set $\{\lambda \in I \mid D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda) \leq \lambda\}$ is nonempty. By Lemma 2, for any $\lambda \in I$, we have $D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda) = D(\mathcal{H}_x^{\lambda_2}, \mathcal{H}_y^{\lambda_2})$. Then, for any λ in I such that $\lambda \geq D(\mathcal{H}_x^{\lambda_2}, \mathcal{H}_y^{\lambda_2})$, we have $D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda) \leq \lambda$ and, for any λ in I such that $\lambda < D(\mathcal{H}_x^{\lambda_2}, \mathcal{H}_y^{\lambda_2})$, we have $D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda) > \lambda$. Hence, in this case, we deduce that $\min\{\lambda \in I \mid D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda) \leq \lambda\}$ is the minimum value of $\mathbb{E}$ which is not less than $D(\mathcal{H}_x^{\lambda_2}, \mathcal{H}_y^{\lambda_2})$. Thus, in this case, we have $\min\{\lambda \in I \mid D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda) \geq \lambda\} = \hat{\mathrm{n}}_{\mathbb{E}}\left(D(\mathcal{H}_x^{\lambda_2}, \mathcal{H}_y^{\lambda_2})\right)$. Furthermore, since $\lambda_1 < D(\mathcal{H}_x^{\lambda_2}, \mathcal{H}_y^{\lambda_2})$, we deduce that $\mathrm{n}_{\mathbb{E}}(\lambda_1) \leq \hat{\mathrm{n}}_{\mathbb{E}}\left(D(\mathcal{H}_x^{\lambda_2}, \mathcal{H}_y^{\lambda_2})\right)$. Thus, in this case, we also have $\min\{\lambda \in I \mid D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda)\} = \max\left\{\mathrm{n}_{\mathbb{E}}(\lambda_1), \hat{\mathrm{n}}_{\mathbb{E}}\left(D(\mathcal{H}_x^{\lambda_2}, \mathcal{H}_y^{\lambda_2})\right)\right\}$.

(3) Let us finally assume that $\lambda_2 < D(\mathcal{H}_x^{\lambda_2}, \mathcal{H}_y^{\lambda_2})$. By Lemma 2, for any $\lambda \in I$, we have $D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda) = D(\mathcal{H}_x^{\lambda_2}, \mathcal{H}_y^{\lambda_2})$. Hence, for any λ in I, we have $D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda) > \lambda$. Hence, in this case the set $\{\lambda \in I \mid D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda) \leq \lambda\}$ is empty.

From the statements given in cases (1), (2), and (3) above, we can affirm that $\{\lambda \in I \mid D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda) \leq \lambda\}$ is nonempty if and only if $D(\mathcal{H}_x^{\lambda_2}, \mathcal{H}_y^{\lambda_2}) \leq \lambda_2$, which completes the proof of Property 3.1. Furthermore, from the statements given in cases (1) and (2), we deduce that, if $D(\mathcal{H}_x^{\lambda_2}, \mathcal{H}_y^{\lambda_2}) \leq \lambda_2$, then we have $\min\left\{\lambda \in I \mid D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda) \leq \lambda\right\} = \max\left\{\mathrm{n}_{\mathbb{E}}(\lambda_1), \hat{\mathrm{n}}_{\mathbb{E}}\left(D(\mathcal{H}_x^{\lambda_2}, \mathcal{H}_y^{\lambda_2})\right)\right\}$, which completes the proof of Property 3.2. □

Definition 4 (stable partition). Let f be any map from $E(T)$ in $\mathbb{E}$, let $u = \{x, y\}$ be any edge in $E(T)$. Let $\mathbb{I} = (\lambda_0, \ldots, \lambda_\ell)$ be a series of real values in $\mathbb{E} \cup \{-\infty\}$ such that $\lambda_0 = -\infty$, $\lambda_\ell = \max\{\lambda \in \mathbb{E}\}$, and, for any i in $\{1, \ldots, \ell\}$, we have $\lambda_i > \lambda_{i-1}$. Let $\mathcal{P}_{\mathbb{I}} = \{]\!]\lambda_0, \lambda_1]\!]_{\mathbb{E}}, \ldots,]\!]\lambda_{\ell-1}, \lambda_\ell]\!]_{\mathbb{E}}\}$. The series $\mathbb{I}$ (resp. the set $\mathcal{P}_{\mathbb{I}}$) is called a *stable bound series (resp. a stable partition) of* $\mathbb{E}$ *for* (f, u) if any element of $\mathcal{P}_{\mathcal{I}}$ is a stable interval for (f, u).

Given a stable bound series $\mathbb{I} = (\lambda_0, \ldots, \lambda_\ell)$, it can be easily seen that the stable partition $\mathcal{P}_{\mathbb{I}}$ is a partition of $\mathbb{E}$. Moreover, given the stable bound series $\mathbb{I} = (\lambda_0, \ldots, \lambda_\ell)$, we denote by $\mathrm{ind}(\mathbb{I})$ the set of indices $\{1, \ldots, \ell\}$.

The following theorem, which is the main result of this section, shows that given a stable bound series, the minimization problem defined by Equation (5) can be solved by considering only the elements of this series rather than all elements of the domain $\mathbb{E}$. Hence, this result is a keystone to provide an efficient algorithm for

computing $\lambda^{\star}_{\mathcal{H}}(\{x,y\})$ with a reduced complexity compared to the naive algorithm (Algorithm 1).

Theorem 5. *Let f be any map from E in $\mathbb{E}$, let $\mathcal{H}$ be the quasi-flat zone hierarchy of G for f, let $u = \{x,y\}$ be any edge in E, and let $\mathbb{I} = (\lambda_0, \ldots, \lambda_\ell)$ be a stable bound series of $\mathbb{E}$ for (f,u). Then, the following statement holds true:*

$$\lambda^{\star}_{\mathcal{H}}(\{x,y\}) = \min\{\max\{n_{\mathbb{E}}(\lambda_{i-1}), \hat{n}_{\mathbb{E}}\left(D(\mathcal{H}^{\lambda_i}_x, \mathcal{H}^{\lambda_i}_y)\right)\} \mid i \in ind(\mathbb{I}), D(\mathcal{H}^{\lambda_i}_x, \mathcal{H}^{\lambda_i}_y) \leq \lambda_i\}. \quad (8)$$

Proof. Let $I_i =]\!]\lambda_{i-1}, \lambda_i]\!]_{\mathbb{E}}$, for any i in $\mathrm{ind}(\mathbb{I}) = \{1, \ldots, \ell\}$. It can be easily seen that $\mathcal{P}_{\mathbb{I}} = \{I_1, \ldots, I_\ell\}$. Since $\mathcal{P}_{\mathbb{I}}$ is a partition of $\mathbb{E}$, we may affirm that:

$$\min\left\{\lambda \in \mathbb{E} \mid D(\mathcal{H}^{\lambda}_x, \mathcal{H}^{\lambda}_y) \leq \lambda\right\} =$$
$$\min\left\{\min\left\{\lambda \in I_i \mid D(\mathcal{H}^{\lambda}_x, \mathcal{H}^{\lambda}_y) \leq \lambda\right\} \mid i \in \mathrm{ind}(\mathbb{I}), \left\{\lambda \in I_i \mid D(\mathcal{H}^{\lambda}_x, \mathcal{H}^{\lambda}_y) \leq \lambda\right\} \neq \emptyset\right\}$$

Since, for any $i \in \{1, \ldots, \ell\}$ the interval $I_i =]\!]\lambda_{i-1}, \lambda_i]\!]_{\mathbb{E}}$ is a stable interval, by Property 1, we deduce that:

$$\min\left\{\lambda \in \mathbb{E} \mid D(\mathcal{H}^{\lambda}_x, \mathcal{H}^{\lambda}_y) \leq \lambda\right\} = \min\{\max\{\mathrm{n}_{\mathbb{E}}(\lambda_{i-1}), \hat{\mathrm{n}}_{\mathbb{E}}\left(D(\mathcal{H}^{\lambda_i}_x, \mathcal{H}^{\lambda_i}_y)\right)\} \mid i \in \mathrm{ind}(\mathbb{I}), D(\mathcal{H}^{\lambda_i}_x, \mathcal{H}^{\lambda_i}_y) \leq \lambda_i\}$$

□

3.3. *Minimization by range*

Let f be any map from E to $\mathbb{E}$, the *range of* f, denoted by $\mathrm{range}(f)$, is the set of values that f can take as its argument varies over E: $\mathrm{range}(f) = \{f(u) \mid u \in E\}$. We denote by $\mathbb{R}_f$ the series $(\lambda_0, \ldots, \lambda_\ell)$ of ordered values in $\mathrm{range}(f) \cup \{-\infty, \max\{\lambda \in \mathbb{E}\}\}$:

(1) $\lambda_0 = -\infty$, and $\lambda_\ell = \max\{\lambda \in \mathbb{E}\}$;
(2) $\{\lambda_i \mid i \in \{1, \ldots, \ell-1\}\} = \mathrm{range}(f) \setminus \{\max\{\lambda \in \mathbb{E}\}\}$; and
(3) for any i in $\{1, \ldots \ell\}$, we have $\lambda_i > \lambda_{i-1}$.

Property 6. Let f be any map from $E(T)$ in $\mathbb{E}$ and let $u = \{x,y\}$ be any edge in $E(T)$. Then, the series $\mathbb{R}_f$ is a stable bound series for (f,u).

Proof. Since $\lambda_0 = -\infty$, since $\lambda_\ell = \max\{\lambda \in \mathbb{E}\}$, and since, for any $i \in \{1, \ldots \ell\}$ we have $\lambda_i < \lambda_{i-1}$, in order to establish Property 6, it is sufficient to prove that, for any i in $\{1, \ldots, \ell\}$, the interval $]\!]\lambda_{i-1}, \lambda_i]\!]_{\mathbb{E}}$ is a stable interval for (f,u). Let i be any element in $\{1, \ldots, \ell\}$. Let λ be any element in $]\!]\lambda_{i-1}, \lambda_i]\!]_{\mathbb{E}}$. Let v be any edge of T which belongs to $f_{\lambda_i}(T)$. By Equation (1), we have $f(v) < \lambda_i$. By definition of $\mathbb{R}_f$, there exists $j \in \{1, \ldots, i-1\}$ such that $f(v) = \lambda_j$. Thus, we have $f(v) < \lambda$, which implies, by Equation (1), that v also belongs to $f_{\lambda_i}(T)$. Furthermore, since $\lambda \leq \lambda_i$, for any edge e such that $f(e) \geq \lambda_i$ we also have $f(e) \geq \lambda$. Hence, by Equation (1),

any edge which does not belong to $f_{\lambda_i}(T)$ does not belong to $f_\lambda(T)$ either. Therefore, we deduce that $w_\lambda(T) = w_{\lambda_i}(T)$. Thus, since $\mathcal{H}$ is the quasi-flat zone hierarchy of f, we also have $\mathcal{H}_x^\lambda = \mathcal{H}_x^{\lambda_i}$ and $\mathcal{H}_y^\lambda = \mathcal{H}_y^{\lambda_i}$, by definition of a quasi-flat zone hierarchy. Hence, the interval $]\!]\lambda_{i-1}, \lambda_i]\!]_{\mathbb{E}}$ is a stable interval for (f, u). □

Corollary 7. *Let f be any map from $E(T)$ in $\mathbb{E}$, let $\mathcal{H}$ be the quasi-flat zone hierarchy of T for f, let $u = \{x, y\}$ be any edge in $E(T)$, and let $\mathbb{R}_f = (\lambda_0, \ldots, \lambda_\ell)$. Then, the following statement holds true:*

$$\lambda^\star_{\mathcal{H}}(\{x,y\}) = \min\{\max\{n_{\mathbb{E}}(\lambda_{i-1}), \hat{n}_{\mathbb{E}}\left(D(\mathcal{H}_x^{\lambda_i}, \mathcal{H}_y^{\lambda_i})\right)\} \mid i \in ind(\mathbb{R}_f), D(\mathcal{H}_x^{\lambda_i}, \mathcal{H}_y^{\lambda_i}) \leq \lambda_i\}. \quad (9)$$

According to Corollary 7, we can compute $\lambda^\star_{\mathcal{H}}(\{x, y\})$ by browsing the values of $\mathbb{R}_f$ in increasing order until a value λ_i such that $D(\mathcal{H}_x^{\lambda_i}, \mathcal{H}_y^{\lambda_i}) \leq \lambda_i$ is found and by setting the value of $\lambda^\star_{\mathcal{H}}(\{x, y\})$ to the maximum between $\mathrm{n}_{\mathbb{E}}(\lambda_{i-1})$ and $\hat{\mathrm{n}}_{\mathbb{E}}\left(D(\mathcal{H}_x^{\lambda_i}, \mathcal{H}_y^{\lambda_i})\right)$. In order to make such process computable, it is necessary to browse the range of f in increasing order. To this end, we propose to store the values of f in a sorted linked list. Algorithm 2 provides a precise description of this process. It can be observed that when the value $\mathrm{p}_{\mathbb{E}}(\lambda^\star_{\mathcal{H}}(\{x, y\}))$ is not yet present in the range of f, the linked list representing this range is updated so that it is ready for the next iteration of the main loop in Method 1. It has to be also noted that in Method 1, the weight of every edge is initialized to the maximal value of $\mathbb{E}$. In other words, the linked list must be initialized in Method 1 with the singleton $\{\max\{\lambda \in \mathbb{E}\}\}$.

Algorithm 2: HGB Minimization by range

Input : A hierarchy $\mathcal{H}$, a weight map f such that $\mathcal{H} = \mathcal{QFZ}(T, f)$, an edge $\{x, y\}$ of T, a linked list L of the values of $\mathbb{R}_f$ in increasing order

Output: The value $\lambda^\star$ such that $\lambda^\star = \lambda^\star_{\mathcal{H}}(\{x, y\})$, the updated linked list L of the values of $\mathbb{R}_f \cup \{\mathrm{p}_{\mathbb{E}}(\lambda^\star)\}$ in increasing order

```
l := L.head; λ := l.value; λ_prev := −∞ ;
while D(H_x^λ, H_y^λ) > λ do
    λ_prev := λ; l := l.next; λ := l.value;
end
λ* := max(n_E(λ_prev), n̂_E(D(H_x^λ, H_y^λ)));
if p_E(λ*) ≠ λ_prev then L.insert(p_E(λ*));
```

As said at the end of Section 3.1, at each iteration of the while loop in Algorithm 2, the dissimilarity between $\mathcal{H}_x^\lambda$ and $\mathcal{H}_y^\lambda$ can be obtained in constant time. The instructions and tests of the while loop in Algorithm 2 are executed at most $|\mathrm{range}(f)|+1$ times and each of these instructions can be made in constant time. Thus, Algorithm 2 runs in $O(|\mathrm{range}(f)|)$ time complexity. It has to be noted

that $|\text{range}(f)|$ is always less than the number of edges of T, which is equal to the number of vertices of T minus one, since T is a tree. This number is, in general, much less than the number of elements of $\mathbb{E}$. Indeed, for reaching a good precision, $\mathbb{E}$ can be chosen as the set of all possible values of the dissimilarity measure D. In such case, the number of elements in $\mathbb{E}$ is in the order of $|\text{range}(w)| \times |V|$. Hence, in this case, the time-complexity is reduced from $O(|\text{range}(w)| \times |V|)$ with Algorithm 1 to $O(|V|)$ with Algorithm 2.

3.4. *Minimization by branch*

In the previous section, we reduce the size of the search space of the minimization defined in Equation (5) by considering the range $\mathbb{R}_f$ of the function f (*i.e.*, a characteristic function of the considered hierarchy $\mathcal{H}$) instead of the set $\mathbb{E}$ of all possible scales of the hierarchy $\mathcal{H}$ (see Corollary 7). In this section, we show that this search space can be further reduced, leading to a third algorithm for computing the value $\lambda^\star_{\mathcal{H}}(\{x, y\})$, given any hierarchy $\mathcal{H}$ and any edge $\{x, y\}$.

In order to obtain this second reduction, we observe in Equation (5) that the only regions of the hierarchy involved in the minimization are those containing x and y. Therefore, while searching for the value $\lambda^\star_{\mathcal{H}}(\{x, y\})$, it is unnecessary to consider a scale of $\mathcal{H}$ (*i.e.*, a value of $\mathbb{R}_f$) at which the regions containing x and y are the same as those at the preceding scale. In other words, rather than considering the scales of $\mathbb{R}_f$ for which there is a global change in the hierarchy, one can focus on the scales for which the change of the hierarchy is local to x and y, *i.e.*, when the change involves a region containing either x or y.

Let f be a any map from E to $\mathbb{E}$ and let $\mathcal{H}$ be the quasi-flat zone hierarchy of f. Let x be any vertex of V and let us denote by $\mathcal{B}_{\mathcal{H}}(x)$ the set which contains every region R of the hierarchy $\mathcal{H}$ such that x belongs to R. The set $\mathcal{B}_{\mathcal{H}}(x)$ is called the *branch of x in $\mathcal{H}$*. The *level* of a region R in $\mathcal{H}$, denoted by $level_{\mathcal{H}}(R)$, is the highest index of a partition that contains R in $\mathcal{H}$. The *(branch) range of $\mathcal{H}$ for x*, denoted by $\text{brange}(f, x)$, is defined as the set that contains the level of every region of the branch of x in $\mathcal{H}$: $\text{brange}(f, x) = \{level_{\mathcal{H}}(R) \mid R \in \mathcal{B}_{\mathcal{H}}(x)\}$. Let $u = \{x, y\}$ be any edge of T. We denote by $\mathbb{R}^u_f$ the series $(\lambda_0, \ldots, \lambda_\ell)$ of ordered values in $\text{brange}(f, x) \cup \text{brange}(f, y) \cup \{-\infty, \max\{\lambda \in \mathbb{E}\}\}$:

(1) $\lambda_0 = -\infty$, and $\lambda_\ell = \max\{\lambda \in \mathbb{E}\}$;
(2) $\{\lambda_i \mid i \in \{1, \ldots, \ell - 1\}\} = \text{brange}(f, x) \cup \text{brange}(f, y) \setminus \{\max\{\lambda \in \mathbb{E}\}\}$; and
(3) for any i in $\{1, \ldots, \ell\}$, we have $\lambda_i > \lambda_{i-1}$.

Property 8. Let f be any map from $E(T)$ in $\mathbb{E}$ and let $u = \{x, y\}$ be any edge in $E(T)$. Then, the series $\mathbb{R}^u_f$ is a stable bound series for (f, u).

Proof. Since $\lambda_0 = -\infty$, since $\lambda_\ell = \max\{\lambda \in \mathbb{E}\}$, and since, for any $i \in \{1, \ldots \ell\}$ we have $\lambda_i < \lambda_{i-1}$, in order to establish Property 8, it is sufficient to prove that, for any i in $\{1, \ldots, \ell\}$, the interval $\rrbracket \lambda_{i-1}, \lambda_i \rrbracket_{\mathbb{E}}$ is a stable interval for (f, u). Let i be

any element in $\{1,\ldots,\ell\}$. Let λ be any element in $]\!]\lambda_{i-1},\lambda_i]\!]_{\mathbb{E}}$. By definition of $\mathbb{R}_f^u$, there exists j in $\{1,\ldots,\ell\}$ such that $\lambda_j = level(\mathcal{H}_x^\lambda)$ (resp. $\lambda_j = level(\mathcal{H}_y^\lambda)$). By definition of the level of a region, we deduce that $\lambda_j \geq \lambda$. Hence, we have $j \geq i$ which implies that $\lambda_j \geq \lambda_i$. Since $\mathcal{H}$ is a hierarchy, we have $\mathcal{H}_x^\lambda \subseteq \mathcal{H}_x^{\lambda_i} \subseteq \mathcal{H}_x^{\lambda_j}$ (resp. $\mathcal{H}_y^\lambda \subseteq \mathcal{H}_y^{\lambda_i} \subseteq \mathcal{H}_y^{\lambda_j}$). By definition of λ_j, we have $\mathcal{H}_x^\lambda = \mathcal{H}_x^{\lambda_j}$ (resp. $\mathcal{H}_y^\lambda = \mathcal{H}_y^{\lambda_j}$). Therefore, we also have $\mathcal{H}_x^\lambda = \mathcal{H}_x^{\lambda_i}$ (resp. $\mathcal{H}_y^\lambda = \mathcal{H}_y^{\lambda_i}$). Thus, $]\!]\lambda_{i-1},\lambda_i]\!]_{\mathbb{E}}$ is a stable interval for (f,u). □

As a direct consequence of Property 8 and Theorem 5, we can state the following corollary which is the basis of a third algorithm for solving efficiently the minimization problem given in Equation (5). The difference with Corollary 7 is that the range of f (f being such that $\mathcal{H} = \mathcal{QFZ}(T,f)$) is replaced by the union of the branch ranges of $\mathcal{H}$ for x and for y.

Corollary 9. *Let f be any map from E in $\mathbb{E}$, let $\mathcal{H}$ be the quasi-flat zone hierarchy of G for f, let $u = \{x,y\}$ be any edge in E, and let $\mathbb{R}_f^u = (\lambda_0,\ldots,\lambda_\ell)$. Then, the following statement holds true:*

$$\lambda^\star_{\mathcal{H}}(\{x,y\}) = \min\{\max\{n_{\mathbb{E}}(\lambda_{i-1}), \hat{n}_{\mathbb{E}}\left(D(\mathcal{H}_x^{\lambda_i},\mathcal{H}_y^{\lambda_i})\right)\} \mid i \in ind(\mathbb{R}_f^u), D(\mathcal{H}_x^{\lambda_i},\mathcal{H}_y^{\lambda_i}) \leq \lambda_i\}. \quad (10)$$

Due to Corollary 9, to compute $\lambda^\star_{\mathcal{H}}(\{x,y\})$, it is sufficient to browse in increasing order the levels of the regions in the branches of x and of y until a value λ_i, such that $D(\mathcal{H}_x^{\lambda_i},\mathcal{H}_y^{\lambda_i}) \leq \lambda_i$, is found. Finally, the value $\lambda^\star_{\mathcal{H}}(\{x,y\})$ is determined as the maximum of $\hat{\mathrm{n}}_{\mathbb{E}}\left(D(\mathcal{H}_x^{\lambda_i},\mathcal{H}_y^{\lambda_i})\right)$ and $\mathrm{n}_{\mathbb{E}}(\lambda_{i-1})$, where $\{\lambda_0,\ldots\lambda_\ell\}$ is equal to $\mathbb{R}_f^{\{x,y\}}$. In order to propose such an algorithm, we need to browse in increasing order the levels of the regions in the branches of x and of y. This can be done with a tree data structure, called a component tree, which represents the hierarchy. The component tree is used for various image processing tasks and is well studied in the field of mathematical morphology (see, *e.g.*, Ref. 37 for its definition on vertex weighted graphs, Ref. 7 for the case of edge-weighted graphs and quasi-flat zone, and Ref. 26 for their generalization to directed graphs). In classification, this tree is often called the dendrogram of the hierarchy.

As any tree, the component tree of $\mathcal{H}$ can be defined as a pair made of a set of nodes and of a binary (parent) relation on the set of nodes. More precisely, the *component tree of* $\mathcal{H}$ is the pair $\mathcal{T}_{\mathcal{H}} = (\mathcal{N}, parent)$ such that $\mathcal{N}$ is the set of all regions of $\mathcal{H}$ and such that a region R_1 in $\mathcal{N}$ is a *parent* of a region R_2 in $\mathcal{N}$ whenever R_1 is a minimal (for inclusion relation) proper superset of R_2. Note that every region in $\mathcal{N}$ has exactly one parent except the region V which has no parent and is called the *root* of the component tree of $\mathcal{H}$. Any region which is not the parent of another one is called a *leaf* of the tree. It can be observed that any singleton of V is a leaf of $\mathcal{T}_{\mathcal{H}}$ and that conversely any leaf of $\mathcal{T}_{\mathcal{H}}$ is a singleton of V.

In order to browse the branch of x in $\mathcal{H}$ from its component tree, it is enough to follow the next steps: (1) start with the node C that is the leaf $\{x\}$, (2) consider

the parent of C, and (3) repeat step (2) until the root is found. Furthermore, it can be observed that the $level_{\mathcal{H}}$ attribute is increasing in the branch of x: for any non-root node C in $\mathcal{N}$, the level of the parent of C is never less than the level of C. Hence, the branch browsing process also allows browsing the branch range of $\mathcal{H}$ for x in increasing order. According to Corollary 9, in order to find the value $\lambda^\star_{\mathcal{H}}(\{x,y\})$, for any edge $\{x,y\}$ of T and any hierarchy $\mathcal{H}$, we have to consider the union of the ranges of $\mathcal{H}$ for x and for y, sorted in increasing order. This can be done by simultaneously browsing in the component tree $\mathcal{T}_{\mathcal{H}}$ the branches of x and of y. Algorithm 3 provides a precise description of a complete algorithm to find $\lambda^\star_{\mathcal{H}}(\{x,y\})$ using such a simultaneous branch browsing.

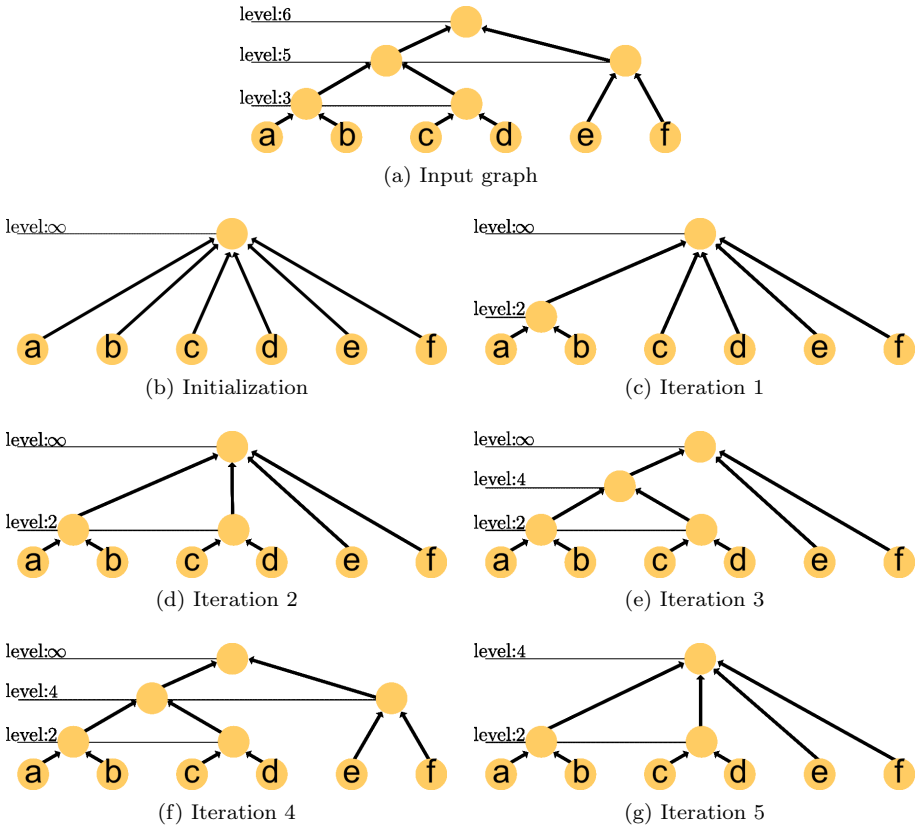
Algorithm 3: HGB Minimization by branch

Input : The component tree $\mathcal{T} = (\mathcal{N}, parent)$ of a hierarchy $\mathcal{H}$, an edge $u = \{x,y\}$ of T, an array $level$ that stores the level of every region of $\mathcal{H}$

Output: The value $\lambda^\star$ such that $\lambda^\star = \lambda^\star_{\mathcal{H}}(\{x,y\})$

1 $C_x := \{x\}$; $C_y := \{y\}$; $\lambda := \min(level[C_x], level[C_y])$; $\lambda_{prev} := -\infty$;
2 **while** $D(C_x, C_y) > \lambda$ **do**
3 $\quad \lambda_{prev} := \lambda$;
4 $\quad \lambda := \min(level[parent[C_x]], level[parent[C_y]])$;
5 $\quad$ **if** $level[parent[C_x]] = \lambda$ **then** $C_x := parent[C_x]$;
6 $\quad$ **if** $level[parent[C_y]] = \lambda$ **then** $C_y := parent[C_y]$;
7 **end**
8 $\lambda^\star := \max(\mathrm{n}_{\mathbb{E}}(\lambda_{prev}), \hat{\mathrm{n}}_{\mathbb{E}}(D(\mathcal{H}^\lambda_x, \mathcal{H}^\lambda_y))$;

Individually, every instruction performed in Algorithm 3 has a constant time complexity. Therefore, in order to establish the overall time complexity of Algorithm 3, it is sufficient to bound the number of iterations of the main loop of Algorithm 3 (Line 2). It can be seen that the instructions and tests of this loop are executed at most $|\mathrm{brange}(f,x)| + |\mathrm{brange}(f,y)|$ times. In the worst case, at every level of the hierarchy the region containing x is merged with a singleton region. Hence, as there are $|V|$ vertices in G, in this case, the branch of x contains $|V|$ regions. Thus, the worst-case time complexity of Algorithm 3 is $O(|V|)$. It can be observed that the worst-case time complexity of Algorithm 3 is the same as the one of Algorithm 2. However, in many practical cases, the component tree of $\mathcal{H}$ is well balanced and each region of $\mathcal{H}$ results from the merging of two regions of (approximately) the same size. Then, if the tree is balanced, the branch of x contains $O(\log_2(|V|))$ nodes and the time-complexity of Algorithm 3 reduces to $O(\log_2(|V|))$ which is a significant improvement compared to Algorithm 2. Such improvement is verified in terms of execution times in Section 4.



(a) Input graph (b) Initialization (c) Iteration 1 (d) Iteration 2 (e) Iteration 3 (f) Iteration 4 (g) Iteration 5

Fig. 4: Tree representations of the quasi-flat zone hierarchies of the graphs of Fig. 2; (g) shows the output hierarchy computed by HGB Method.

3.5. *Quasi-flat zone hierarchy algorithms*

In this section, we focus on Lines 3 and 6 of Method 1, that is, on computing the quasi-flat zone hierarchy of a weight map f. This computation is repeated at every iteration of the method (*i.e.*, for every edge of the tree T). Hence, finding an efficient way to perform this task in the context of Method 1 presents a high speedup potential.

The straightforward implementation of Lines 3 and 6 of Method 1 consists of computing, at each iteration, the quasi-flat zone hierarchy of f using an efficient algorithm such as the one presented in Ref. 20. Provided that the edges of T can be sorted in linear time, which, in our case, can be done with a counting sort algorithm, the time-complexity of this algorithm is quasi-linear $O(|V|+\alpha(|V|))$ (where α is the extremely slowly growing inverse of the single-valued Ackermann function). Hence, in this case, the overall time-complexity of Lines 3 and 6 is $O(|V|^2 + |V|\alpha(|V|))$ since the quasi-flat zone hierarchy computation is repeated exactly $|E(T)| = |V|-1$

times.

However, from one iteration of the main loop of Method 1 to the next one, only the weight of one edge of the graph is updated and therefore most parts of the component tree remain unchanged (see, for instance, Fig. 4). Therefore, an important speedup can be obtained if we avoid to recompute from scratch the whole component tree at each iteration. In order to avoid this recomputation, we need to rely on an algorithm that only updates the part of the component tree which is affected by the single weight update considered at the present iteration. In this section, we propose such an algorithm that is referred to as an incremental quasi-flat zone update algorithm.

The presented incremental quasi-flat zone update algorithm relies on works done for parallel computation of component trees presented in Refs. 40, 15, 9, 3. In these articles, the authors present algorithms to merge the component trees of two disjoint (adjacent) image blocks in order to obtain the component tree of the image consisting of these two blocks. We can adapt these algorithms (in particular Algorithm 6 in Ref. 15) into an incremental quasi-flat zone update algorithm. At each iteration of Line 3 in Method 1, the weight of the edge u is decreased from its initial value $\max\{\lambda \in \mathbb{E}\}$ to its final value resulting from the minimization of Equation (5). This means that, before we decrease the weight of $u = \{x, y\}$, the components containing x and y in the tree were disjoint (up to level $\max\{\lambda \in \mathbb{E}\}$). We can adapt the algorithm proposed by Ref. 15, in order to merge these disjoint parts of the tree and update the tree only on the components containing x and y, thus avoiding the need to recompute the whole hierarchy at every iteration of Method 1.

Algorithm 4 gives a precise description of this quasi-flat zone update algorithm given the component tree $\mathcal{T}$ of a hierarchy $\mathcal{H}$ which is the quasi-flat zone hierarchy of a weight map f, the edge $u = \{x, y\}$ whose weight must be decreased, and the value λ which corresponds to the decreased weight of u. The algorithm first identifies the part of the tree which must be modified, namely the components containing x and y at levels higher than λ (Line 1). Then, the tree representation of the components containing x and y at higher level is built (Lines 2 to 14) by either merging existing nodes (Line 10), creating new parenthood relation between existing nodes (Lines 7 and 11), or creating new nodes (Line 2). To perform these tasks, Algorithm 4 relies on four auxiliary functions.

- Given a vertex x of V and a level λ in $\mathbb{E}$, $findTransition(x, \lambda)$ returns a node n which is the ancestor of the node representing $\{x\}$ such that (i) $level[n] \leq \lambda$, and (ii) $level[parent[n]] > \lambda$. This operation is performed by traversing upward the branch of $\mathcal{T}$ containing x, starting at the node $n = \{x\}$ and ending when a node n satisfying (i) and (ii) is found.
- Given two nodes c_1 and c_2 of $\mathcal{T}$ and a value λ in $\mathbb{E}$, $node(c_1, c_2, \lambda)$ creates a node n at level λ which becomes the parent of c_1 and of c_2.
- Given two nodes c_1 and c_2 of $\mathcal{T}$, $attach(c_1, c_2)$ sets c_1 to be the parent of c_2.

- Given two nodes c_1 and c_2, $merge(c_1, c_2)$ calls $merge(c_2, c_1)$ if c_2 has more children than c_1, otherwise it sets the parent of the children of c_2 to be c_1, and it returns c_1.

Algorithm 4: Incremental quasi-flat zone hierarchy update

Input : The component tree $\mathcal{T} = (\mathcal{N}, parent)$ of the hierarchy $\mathcal{H}$ which is the quasi-flat zone hierarchy of a map f, an array *level* that stores the level of every node of $\mathcal{T}$ (*i.e.*, every region of $\mathcal{H}$), an edge $u = \{x, y\}$ of T, the value λ at which the weight of the edge u must be decreased.

Output: The updated component tree $\mathcal{T}$, which is the component tree of the quasi-flat zone hierarchy of the updated map f (*i.e.*, the map f' such that $f'(v) = f(v)$ for any $v \neq u$ and $f'(u) = \lambda$)

```
1  c1 := findTransition(x, λ) ; c2 := findTransition(y, λ) ;
2  n := node(c1, c2, n_E(λ)) ;
3  c1 := parent[c1]; c2 := parent[c2];
4  do
5  |  if level[c2] < level[c1] then swap(c1, c2);
6  |  if level[c1] < level[c2] then
7  |  |  attach(c1, n) ;
8  |  |  n := c1; c1 := parent[c1] ;
9  |  else
10 |  |  n' := merge(c1, c2) ;
11 |  |  attach(n', n) ;
12 |  |  n := n'; c1 := parent[c1]; c2 := parent[c2];
13 |  end
14 while p1 ≠ p2 ;
```

Algorithm 4 modifies the tree structure in the following manner: first, given an edge $u = \{x, y\}$ of decreased weight λ, it starts from the singleton components $\{x\}$ and $\{y\}$. Then, *findTransition* identifies the nodes c_1 and c_2 associated to the components $\mathcal{H}_x^\lambda$ and $\mathcal{H}_y^\lambda$, respectively. A node n is created to represent the union of theses components (Line 2). Then, the do-while loop (Lines 5 to 14) traverses simultaneously the branches containing x and y from the nodes c_1 and c_2, identifying the ancestors of these nodes, and updating the parenthood relationships along these branches. At each iteration, the two nodes are merged if they have the same level (Line 10) or, if one has a level less than the other, the one of highest level becomes the parent of the one of lowest level (Line 7). This is repeated until a common ancestor is found. Consequently, only the components containing x and y are involved in the update algorithm and we do not need to recompute a whole

hierarchy at every iteration.

It can be seen that Algorithm 4 involves only constant time operations performed on the nodes corresponding to the branch containing x and y. Therefore, following the discussion at the end of Section 3.4, in the worst case, Algorithm 4 runs in $O(|V|)$ time complexity and in the case where the tree $\mathcal{T}$ is balanced it runs in $O(\log_2(|V|))$ time complexity. Hence, in the worst case, using Algorithm 4, the overall time-complexity of Line 3 in Method 1 in Method 1 is $O(|V|^2)$, whereas in the more favorable case where the tree remains balanced, this complexity reduces to $O(|V|\log_2|V|)$.

4. Assessments

The experiments reported in this section aim at measuring and comparing the execution times of all the variations of the algorithms proposed in the previous sections for the HGB method. The experimental set-up is first presented in Section 4.1 and then the experimental results are given in Section 4.2.

4.1. *Experimental set-up*

As we have three variations for the minimization step (Line 4 in Method 1), namely Algorithms 1, 2 and 3, and two variations for the quasi-flat zone hierarchy computation (Lines 3 and 6 in Method 1), namely the non-incremental one (based on Ref. 20) and the incremental one (based on Algorithm 4), the total number of all the combinations is six. Hence, in total, we study the execution times of these six variations.

All the algorithms were implemented in C and executed on a computer with a 3.2 GHz CPU, 8GB RAM on Ubuntu Linux 16.04.

In order to cope with realistic situations, we use the Berkeley Segmentation Dataset (BSDS) proposed by Ref. 1 for our experiments. This dataset consists of 500 natural images of size 321×481 pixels and is very popular in image segmentation experiments.

A first assessment consists of measuring the execution times of the six presented variations when applied to one of the images from the BSDS dataset. The chosen image is shown in Fig. 5 (a) and the result of any of the six variations is shown in Fig. 5 (b) in the form of a saliency map. This image was chosen because of its textured aspect which leads to hierarchies with a high number of regions and scales, hence exploiting the ability of the algorithms to deal with a high number of regions and of levels (5218 levels). This experiment is designed in order to assess the gain achieved from switching from the least efficient variation of Method 1 to the most efficient one.

A second assessment is designed in order to assess the scalability of the most efficient variation of Method 1 when applied to a whole dataset of images representing a wider variety of situations that can be encountered in computer vision tasks. This second experiment consists of measuring the execution times taken by

(a) Input image (b) Hierarchical segmentation

Fig. 5: Image used for the algorithm assessment and the resulting HGB hierarchy represented as a saliency map.

the fastest variation of Method 1 on the full BSDS dataset. The fastest variation to compute the result of Method 1 is determined from the first assessment and consists of using Algorithm 3 and Algorithm 4.

Reported execution times result from repeating ten times the execution of a same method on a same image and considering the average time taken by these ten executions.

4.2. *Experimental results*

Table 1 shows the execution times taken by each of the six variations of Method 1 when applied to the image shown in Fig. 5 (a). In the table, the first column labeled *QFZ algorithm* refers to the type of algorithm used to construct the QFZ hierarchy: it can be either the non-incremental one based on Ref. 20 or the incremental one, namely Algorithm 4, based on Refs. 15, 40. The second column refers to the algorithm which is used for performing the minimization described in Equation (5): it can be either the naive algorithm (Algorithm 1), the minimization by range algorithm (Algorithm 2) or the minimization by branch algorithm (Algorithm 3). The third column presents the overall execution times of the six variations which include the time for computing quasi-flat zone hierarchies (fourth column) and for computing the result of the minimization described by Equation (5) (fifth column).

As we can observe in Table 1, the total execution time using the non-incremental

Table 1: Execution times from the image of Fig. 5(a) (321×481 pixels). The resulting hierarchy contains 5218 levels.

QFZ Algorithm	**Minimization Algorithm**	**Execution times (seconds)**		
		Total	**QFZ**	**Minimization**
Non-Incremental	Algorithm 1	14666.08	13186.31	1479.56
	Algorithm 2	13392.51	13375.29	17.02
	Algorithm 3	13166.25	13165.54	0.49
Incremental	Algorithm 1	1487.96	0.13	1487.75
	Algorithm 2	15.42	0.13	15.21
	Algorithm 3	0.49	0.10	0.32

QFZ hierarchy construction and any minimization algorithm leads into very prohibitive times of over four hours. However, most of this time is consumed on the QFZ hierarchy construction (over three hours). For the minimization step, Algorithm 1 is the least efficient taking around 24 minutes. Then, Algorithm 2 using the range minimization leads to 17 seconds of execution. Finally, Algorithm 3 is the fastest algorithm for the minimization step with less than one second. When we use the incremental QFZ hierarchy construction, the time spent on updating the hierarchical tree takes only 0.1 seconds. This, together with Algorithm 3 for the minimization step, leads to a total execution time of 0.49 seconds, which is our most efficient variation. From these results, we can conclude that it is only possible to compute the HGB method in user time with the help of both the incremental QFZ hierarchy construction and the minimization by branch (Algorithm 3). Observe that, at each iteration of the main loop of Method 1, the QFZ construction takes approximatively 0.086 seconds in average with the non-incremental (original) version, whereas, with the incremental version, the handling of the QFZ hierarchy takes, at every iteration, 0.00000084 seconds in average.

Figure 6 shows the distribution of the execution times of our most efficient variation of Method 1 applied to all images in the BSDS dataset. The average execution time over the dataset is 0.47 seconds with a standard deviation of 0.09 seconds. This confirms that that our most efficient variation of Method 1, namely the implementation of Method 1 with Algorithm 3 and Algorithm 4, runs in user time whatever the considered image from the BSDS dataset.

5. Conclusions

In this article, we investigated the HGB method proposed in Ref. 13 with the aim of developing exact and time-efficient algorithms for its implementation on images. We focused on the two main steps of the HGB method for improving efficiency: (i) the minimization involved in Equation (5), and (ii) the computation of the quasi-flat zones hierarchies. Concerning (i), we presented a general framework which allows reducing the search space involved in the minimization problem as

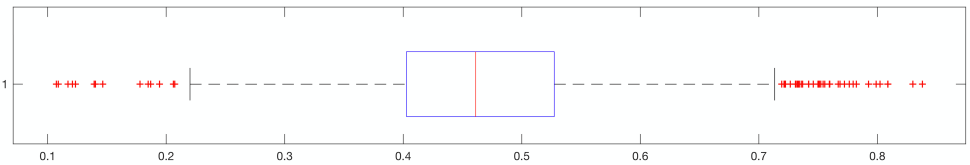


Fig. 6: Box and Whisker plot for execution times on BSDS dataset. On the horizontal axis the times are given in seconds.

shown by Theorem 5. We considered two applications of this framework leading to two algorithms which both improve the efficiency, in terms theoretical of time complexity and of practical running times, compared to a naive algorithm for solving the minimization problem. Furthermore, due to the proposed framework, the proof of correctness for each of these two algorithms (namely Corollaries 7 and 9) is provided. In order to compute efficiently the quasi-flat zone hierarchy (ii), we considered a non-incremental and an incremental algorithm based on Ref. 20 and on Ref. 15, respectively. Even if the worst-case complexities of these two algorithms are comparable, the running times of the HGB method are significantly decreased when the incremental algorithm is used instead of the non-incremental one. Overall, on images from the standard BSDS dataset, the least efficient strategy that we proposed computes the result of the HGB method in more than four hours whereas the most efficient one takes about half a second.

Furthermore, we would like to emphasize that the framework presented in this article leads to a better understanding of the minimization equation which is at the heart of the HGB method. In particular, it opens doors towards modifications of this equation which could allow us to significantly improve the quality of the resulting hierarchies. Among others, in future works, we may be able to propose fast and exact algorithms to compute the results of HGB method using different dissimilarity measures than the one of Felzenszwalb-Huttenlocher. In this direction, we may for instance consider the criteria presented in Ref. 22 or in Ref. 24 which are more complex than the one of Felzenszwalb and Huttenlocher. First results in this direction are encouraging (see Refs. 12, 14). Moreover, in order to robustify the results in practice, we may also consider replacing the minimum of the solutions in stable intervals by the maximum, the median or a percentile of the solutions in the stable intervals. These interesting questions are beyond the scope of the present article and represent interesting research topic for our future works.

Acknowledgements

The research leading to these results has received funding from the French Agence Nationale de la Recherche, grant agreement ANR-15-CE40-0006 (CoMeDiC), the Brazilian Federal Agency of Support and Evaluation of Postgraduate Education (program CAPES/PVE: grant 064965/2014-01), the Peruvian agency Consejo Na-

cional de Ciencia, Tecnología e Innovación Tecnológica CONCYTEC (contract N 101-2016-. FONDECYT-DE). The first author would like to thank Brazilian agencies CNPq and CAPES and Peruvian agency CONCYTEC for the financial support during his thesis.

References

1. P. Arbelaez, M. Maire, C. Fowlkes and J. Malik, Contour detection and hierarchical image segmentation, *TPAMI* (2011) 898–916.
2. S. Beucher, Watershed, hierarchical segmentation and waterfall algorithm, in *ISMM* (1994) pp. 69–76.
3. E. Carlinet and T. Graud, A comparative review of component tree computation algorithms, *IEEE Transactions on Image Processing* **23**(9) (2014) 3885–3895.
4. E. J. Y. Cayllahua Cahuina, J. Cousty, Y. Kenmochi, A. De Albuquerque Araújo and G. Cámara-Chávez, Algorithms for hierarchical segmentation based on the Felzenszwalb-Huttenlocher dissimilarity, in *International Conference on Pattern Recognition and Artificial Intelligence* (2018)
5. J. Cousty and L. Najman, Incremental algorithm for hierarchical minimum spanning forests and saliency of watershed cuts, in *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing* (2011) pp. 272–283.
6. J. Cousty, L. Najman, Y. Kenmochi and S. Guimarães, Hierarchical segmentations with graphs: Quasi-flat zones, minimum spanning trees, and saliency maps, *JMIV* (2017) 1–22.
7. J. Cousty, L. Najman and B. Perret, Constructive links between some morphological hierarchies on edge-weighted graphs, in *ISMM* (2013) pp. 135–146.
8. P. F. Felzenszwalb and D. P. Huttenlocher, Efficient graph-based image segmentation, *IJCV* (2004) 167–181.
9. M. Götz, G. Cavallaro, T. Graud, M. Book and M. Riedel, Parallel computation of component trees on distributed memory machines, *IEEE Transactions on Parallel and Distributed Systems* (2018) 1–1.
10. L. Guigues, J. P. Cocquerez and H. Le Men, Scale-sets image analysis, *IJCV* (2006) 289–317.
11. S. Guimarães, Y. Kenmochi, J. Cousty, Z. Patrocinio Jr. and L. Najman, Hierarchizing graph-based image segmentation algorithms relying on region dissimilarity - the case of the Felzenszwalb-Huttenlocher method, *Math. Morphol. Theory Appl.* (2017) 1–22.
12. S. J. F. Guimarães and J. Patrocínio, Zenilton K.G., A graph-based hierarchical image segmentation method based on a statistical merging predicate, in A. Petrosino (ed.), *Image Analysis and Processing - ICIAP 2013*, *Lecture Notes in Computer Science*Vol. 8156 (Springer Berlin Heidelberg, 2013) pp. 11–20.
13. S. Guimarães, J. Cousty, Y. Kenmochi and L. Najman, A hierarchical image segmentation algorithm based on an observation scale, in *SSPR* (2012) pp. 116–125.
14. S. J. F. Guimarães, Z. K. G. do Patrocínio Jr, Y. Kenmochi, J. Cousty and L. Najman, Hierarchical image segmentation relying on a likelihood ratio test, in *Image Analysis and Processing–ICIAP 2015* (Springer International Publishing, 2015) pp. 25–35.
15. J. Havel, F. Merciol and S. Lefèvre, Efficient tree construction for multiscale image representation and processing, *JRTIP* (2016) 1–18.
16. D. Hoiem, A. A. Efros and M. Hebert, Recovering occlusion boundaries from an image, *International Journal of Computer Vision* **91**(3) (2011) 328–346.
17. R. Marfil, L. Molina-Tanco, A. Bandera, J. Rodríguez and F. Sandoval, Pyramid segmentation algorithms revisited, *Pattern Recognition* **39**(8) (2006) 1430 – 1451.

18. F. Meyer, The dynamics of minima and contours, in *ISMM* (1996) pp. 329–336.
19. F. Meyer and P. Maragos, Morphological scale-space representation with levelings, in *Scale-Space Theories in Computer Vision* (1999) pp. 187–198.
20. L. Najman, J. Cousty and B. Perret, Playing with Kruskal: algorithms for morphological trees in edge-weighted graphs, in *ISMM* (2013) pp. 135–146.
21. L. Najman and M. Schmitt, Geodesic saliency of watershed contours and hierarchical segmentation, *TPAMI* (1996) 1163–1173.
22. R. Nock and F. Nielsen, Statistical region merging, *IEEE Transactions on pattern analysis and machine intelligence* **26**(11) (2004) 1452–1458.
23. T. Pavlidis, *Structural Pattern Recognition* (Springer, 1977).
24. B. Peng, L. Zhang and D. Zhang, Automatic image segmentation by dynamic region merging, *IEEE Transactions on Image Processing* **20**(12) (2011) 3592–3605.
25. B. Perret, J. Cousty, S. J. Guimares and D. S. Maia, Evaluation of hierarchical watersheds, *IEEE Transactions on Image Processing* **27**(4) (2018) 1676–1688.
26. B. Perret, J. Cousty, O. Tankyevych, H. Talbot and N. Passat, Directed connected operators: asymmetric hierarchies for image filtering and segmentation, *TPAMI* (2015) 1162–1176.
27. B. Perret, J. Cousty, J. C. R. Ura and S. J. F. Guimarães, Evaluation of morphological hierarchies for supervised segmentation, in J. A. Benediktsson, J. Chanussot, L. Najman and H. Talbot (eds.), *Mathematical Morphology and Its Applications to Signal and Image Processing* (Springer International Publishing, Cham, 2015) pp. 39–50.
28. M. Pham, S. Lefèvre, E. Aptoula and L. Bruzzone, Recent Developments from Attribute Profiles for Remote Sensing Image Classification, in *International Conference on Pattern Recognition and Artificial Intelligence* (2018)
29. J. Pont-Tuset, P. Arbeláez, J. T. Barron, F. Marques and J. Malik, Multiscale combinatorial grouping for image segmentation and object proposal generation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **39**(1) (2017) 128–140.
30. J. Pont-Tuset and F. Marqués, Supervised assessment of segmentation hierarchies, in *European Conference on Computer Vision (ECCV)* (2012)
31. J. Pont-Tuset and F. Marques, Upper-bound assessment of the spatial accuracy of hierarchical region-based image representations, in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2012) pp. 865–868.
32. J. Pont-Tuset and F. Marques, Supervised evaluation of image segmentation and object proposal techniques, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **38**(7) (2016) 1465–1478.
33. J. F. Randrianasoa, C. Kurtz, P. Gancarski, E. Desjardin and N. Passat, Intrinsic quality analysis of binary partition trees, in *International Conference on Pattern Recognition and Artificial Intelligence (ICPRAI)* (2018)
34. Z. Ren and G. Shakhnarovich, Image segmentation by cascaded region agglomeration, in *IEEE Conference on Computer Vision and Pattern Recognition* (June 2013) pp. 2011–2018.
35. P. Salembier and S. Foucher, Optimum graph cuts for pruning binary partition trees of polarimetric sar images, *IEEE Transactions on Geoscience and Remote Sensing* (2016) 5493–5502.
36. P. Salembier and L. Garrido, Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval, *TIP* (2000) 561–576.
37. P. Salembier, A. Oliveras and L. Garrido, Antiextensive connected operators for image and sequence processing, *TIP* (1998) 555 – 570.
38. P. Soille, Constrained connectivity for hierarchical image decomposition and simplification, *TPAMI* (July 2008) 1132–1145.

39. J.-H. Syu, S.-J. Wang and L. Wang, Hierarchical image segmentation based on iterative contraction and merging, *TIP* (2017) 2246 – 2260.
40. M. H. Wilkinson, H. Gao, W. H. Hesselink, J.-E. Jonker and A. Meijster, Concurrent computation of attribute filters on shared memory parallel machines, *TPAMI* (2008) 1800–1813.
41. Y. Xu, E. Carlinet, T. Géraud and L. Najman, Hierarchical segmentation using tree-based shape spaces, *IEEE transactions on pattern analysis and machine intelligence* **39**(3).
42. Q. Yan, L. Xu, J. Shi and J. Jia, Hierarchical saliency detection, in *2013 IEEE Conference on Computer Vision and Pattern Recognition* (2013) pp. 1155–1162.
43. Q. Zhao, Segmenting natural images with the least effort as humans, in M. W. J. Xianghua Xie and G. K. L. Tam (eds.), *Proceedings of the British Machine Vision Conference (BMVC)* (BMVA Press, September 2015) pp. 110.1–110.12.