An algebraic framework for out-of-core hierarchical segmentation algorithms

Jean Cousty[1], Benjamin Perret[1], Harold Phelippeau[2], Stela Carneiro[1], Pierre Kamlay[2], and Lilian Buzer[1]

[1] LIGM, Univ Gustave Eiffel, CNRS, ESIEE Paris, F-77454 Marne-la-Vallée
[2] Thermo Fisher Scientific, Bordeaux, France

Abstract. Binary partition hierarchies and minimum spanning trees are key structures for numerous hierarchical analysis methods, as those involved in computer vision and mathematical morphology. In this article, we consider the problem of their computation in an out-of-core manner, *i.e.*, by minimizing the size of the data structures that are simultaneously needed at the different computation steps. Out-of-core algorithms are necessary when the data are too large to fit entirely in the main memory of the computer, which can be the case with very large images in 2-, 3-, or higher dimension space. We propose a new algebraic framework composed of four main operations on hierarchies: edge-addition, select, insert, and join. Based on this framework, we propose and establish the correctness of an out-of-core calculus for binary partition hierarchies and for minimum spanning trees. First applications to image processing suggest the practical efficiency of this calculus.

1 Introduction

Image segmentation, one of the oldest problems in computer vision, consists in partitioning an image into meaningful regions that can be used to perform higher-order tasks. However, the objects of interest do not all appear at the same scale and can be nested within each other making segmentation a ill-posed problem. In this article, we are interested in the more general problem of hierarchical (or multi-scale) segmentation: find a series of partitions ordered from fine to coarse describing how the finer details are grouped to form higher level regions. This problem generalizes the one of hierarchical clustering, studied in classification, with the additional benefit of considering class-connectivity in a graph framework [2]. Nowadays, state of the art image analysis procedures involving segmentation [4, 11] include two steps performed independently: i / learning contour cues and regional attributes with machine or deep learning strategies; and ii / computing and processing hierarchical segmentation trees. One of the trends in the computer vision community is the integration of these two tasks with for instance the introduction of hierarchy processing layers [1] into deep learning architectures for use in an end-to-end fashion. Binary partition hierarchies [17] built from an altitude ordering (a total order on the pairs of adjacent pixels or vertices) and minimum spanning trees [3,12] constitute key structures for numerous hierarchical analysis methods.

While the algorithms for building and processing hierarchical segmentations are well established in the regular case of a single image of a standard size, there is a lack of efficient scalable algorithms. Scalability is needed to analyze large images at native resolution or large datasets. Indeed, giga- or tera-bytes images, which become common with the improvements in sensor resolution, cannot fit within the main memory of a computer and recent (deep) learning methods require browsing several times datasets of millions of images. The classical sequential hierarchy algorithms are not adapted to these situations and therefore need to be completely redesigned [10,5,7,6] to cope with these situations.

In [10,5,7], the authors investigate parallel algorithms to compute the min and max trees – hierarchical structures used for various applications, such as attribute filtering and segmentation – of terabytes images. In [6], the computation of minimum spanning trees of streaming images is considered. In [9], the authors investigate the parallel computation of quasi-flat zones hierarchies. From a high-level point of view, these approaches work independently on small pieces of the space, "join" the information found on adjacent pieces, and "propagate" (or "insert") this joint information into other pieces.

In this article, we envision the problem of computing binary partition hierarchies (and associated minimum spanning trees) under the out-of-core constraint, that is minimizing the size of the data structures loaded simultaneously into the principal memory of the computer. In other words, out-of-core algorithms target the processing of data or images that are too large to fit into the main memory at once without necessarily using any parallelization. With respect to these objectives, our main contributions in this article are the following:

- the formalization of a representation of a hierarchy of partitions, called a distribution, that is suited to out-of-core hierarchical analysis (Section 2);
- the introduction and the algebraic study of a fundamental external binary operation on hierarchies called edge-addition. This operation allows us to provide new characterizations of binary partition hierarchies and to introduce the notion of a partial binary partition hierarchy that is useful in the context of distributions (Section 3);
- three operations on hierarchies called select, join, and insert (Sections 2 and 4) to handle distributions;
- an out-of-core calculus for (distributions of) binary partition hierarchies and minimum spanning trees (over a causal partition) of the space (Section 5); this calculus only involves the three introduced operations applied to partial hierarchies and it requires $O(K)$ calls to these operations, where K is the number of parts onto which the data structures are distributed;
- a discussion and a proof of concept of the efficiency of this calculus to design out-of-core algorithms for binary partition hierarchies and minimum spanning trees (Section 6).

Due to the space limitation in this article, formal proofs of the properties are omitted and will be given in a forthcoming extended version.

2 Distributed hierarchies of partitions: select operation

In this section, we first remind the definitions of a partial partition and of a hierarchy of partial partitions [15, 16]. Then, we introduce the select operation. Intuitively, this operation consists in "selecting" the part of a given hierarchy made of the regions which hit a given set. Finally, we define the notion of a distribution of a hierarchy over a partition of the space. Such distribution is a set of hierarchies where each element is the result of "selecting" the part of the given hierarchy associated with one of the class of the given partition. These notions allow one to define a component forest and a border tree as presented in [10, 5] and to set up the formal problem that we tackle in this article.

Let V be a set. A *(partial) partition of* V is a set of pairwise disjoint subsets of V. Any element of a partition is called a *region* of this partition. The *support (or ground)* of a partition $\mathbf{P}$, denoted by $gr(\mathbf{P})$, is the union of the regions of $\mathbf{P}$. A partition whose support is V is called a *complete partition of* V. Let $\mathbf{P}$ and $\mathbf{Q}$ be two partitions of V. We say that $\mathbf{Q}$ is a *refinement of* $\mathbf{P}$ if any region of $\mathbf{Q}$ is included in a region of $\mathbf{P}$. A *hierarchy on* V is a series $(\mathbf{P}_0, \ldots, \mathbf{P}_\ell)$ of partitions of V such that, for any λ in $\{0, \ldots, \ell - 1\}$, the partition $\mathbf{P}_\lambda$ is a refinement of $\mathbf{P}_{\lambda+1}$. Let $\mathcal{H} = (\mathbf{P}_0, \ldots, \mathbf{P}_\ell)$ be a hierarchy. The integer ℓ is called the *depth of* $\mathcal{H}$ and, for any λ in $\{0, \ldots, \ell\}$, the partition $\mathbf{P}_\lambda$ is called the *λ-scale of* $\mathcal{H}$. In the following, if λ is an integer in $\{0, \ldots, \ell\}$, we denote by $\mathcal{H}[\lambda]$ the λ-scale of $\mathcal{H}$. For any λ in $\{1, \ldots, \ell\}$, any region of the λ-scale of $\mathcal{H}$ is also called a *region of* $\mathcal{H}$. The hierarchy $\mathcal{H}$ is *complete* if $\mathcal{H}[0] = \{\{x\} \mid x \in V\}$ and if $\mathcal{H}[\ell] = \{V\}$. The hierarchy $\mathcal{H}$ is *binary* if, for any $\lambda \in \{1, \ldots, \ell\}$, we have $|\mathbf{P}_\lambda| \in \{|\mathbf{P}_{\lambda-1}| - 1, |\mathbf{P}_{\lambda-1}|\}$. We denote by $\mathcal{H}_\ell(V)$ the set of all hierarchies on V of depth ℓ, by $\mathbf{P}(V)$ the set of all partitions on V, and by $2^{|V|}$ the set of all subsets of V.

In the following, the symbol ℓ stands for any positive integer.

Let V be a set. The operation *sel* is the map from $2^{|V|} \times \mathbf{P}(V)$ to $\mathbf{P}(V)$ which associates to any subset X of V and to any partition $\mathbf{P}$ of V the subset $\mathrm{sel}(X, \mathbf{P})$ of $\mathbf{P}$ which contains every region of $\mathbf{P}$ that contains an element of X. The operation *select* is the map from $2^{|V|} \times \mathcal{H}_\ell(V)$ in $\mathcal{H}_\ell(V)$ which associates to any subset X of V and to any hierarchy $\mathcal{H}$ on V the hierarchy $\mathrm{select}(X, \mathcal{H}) = (\mathrm{sel}(X, \mathcal{H}[0]), \ldots, \mathrm{sel}(X, \mathcal{H}[\ell]))$.

Definition 1 (Distributed hierarchy) *Let V be a set, let $\mathbf{P}$ be a complete partition on V and let $\mathcal{H}$ be a hierarchy on V. The* distribution of $\mathcal{H}$ over $\mathbf{P}$ *is the set $\{select(R, \mathcal{H}) \mid R \in \mathbf{P}\}$.*

The operation *select* and the notion of a distribution of a hierarchy are illustrated in Fig. 1. The following property asserts that the distribution of any hierarchy $\mathcal{H}$ over a partition of the space is indeed a representation of this hierarchy, which means that one can retrieve $\mathcal{H}$ from its distribution.

Property 2 (reconstruction) *Let V be a set, let $\mathbf{P}$ be a complete partition on V, let $\mathcal{H}$ be a hierarchy on V of depth ℓ, and let δ be the distribution of $\mathcal{H}$ over $\mathbf{P}$. Then, for any λ in $\{0, \ldots, \ell\}$, we have $\mathcal{H}[\lambda] = \cup\{\mathcal{D}[\lambda] \mid \mathcal{D} \in \delta\}$.*

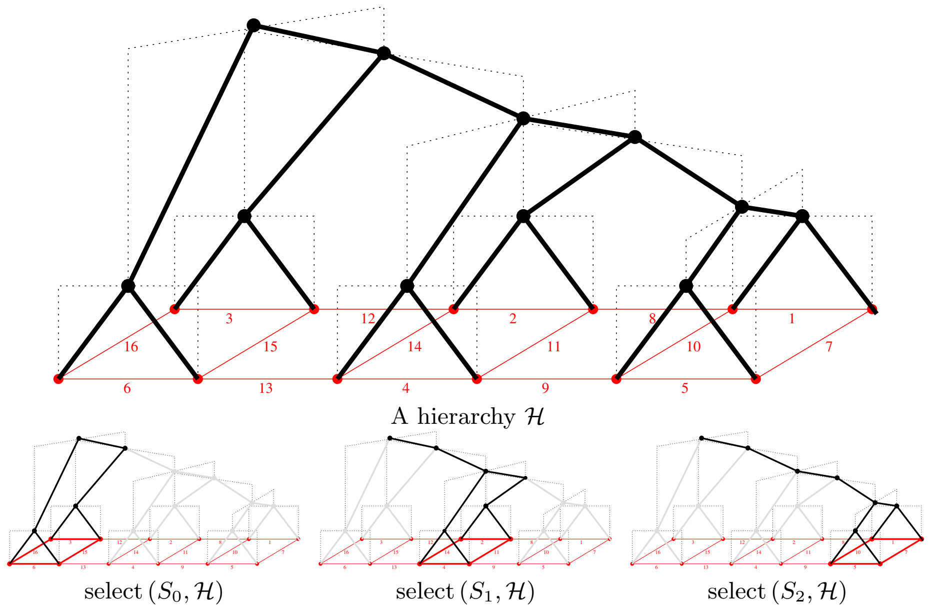
3
12
2
8
1
16
15
14
11
10
7
6
13
4
9
5
A hierarchy $\mathcal{H}$
select $(S_0, \mathcal{H})$
select $(S_1, \mathcal{H})$
select $(S_2, \mathcal{H})$

Fig. 1. Illustration of a distribution of a hierarchy. The hierarchies are represented in black-bold in the form of trees. First row: the binary partition hierarchy of the weighted graph depicted in red; and second row: its distribution over the partition $\{S_0, S_1, S_2\}$, where S_0 (resp. S_1, S_2) contains the four leftmost (resp. centermost, rightmost) vertices of the graph.

3 Binary partition hierarchies: edge-addition operations

The binary partition hierarchy plays a central role when one has to deal with hierarchies of partitions, notably in the framework of mathematical morphology [3, 12, 3]. It allows one to obtain the set representation of a hierarchy from its saliency map or from its ultrametric distance representations [3, 13]. It can also be efficiently post-processed to obtain other hierarchies useful in image analysis applications like the quasi-flat zones and the constrained connectivity hierarchies [18], the hierarchical watersheds [13], the various HGB hierarchies [8], and to simplify hierarchies by removing non-significant regions [14] or to obtain marker-based segmentations of an image [17]. When the binary partition hierarchy is built from the edge-weights of a graph, its regions are in bijection with the vertices and edges of a minimum spanning tree of this graph [3, 12] and thus embeds this minimum spanning tree.

In this section, we provide a new definition of this hierarchy based on an (elementary) external binary operation on hierarchies. We investigate the algebraic properties of this operation. As shown in Section 5, this allows us to design calculus to obtain distributions of binary partition hierarchies.

The binary partition hierarchy depends on an ordering of the edges of a graph that structures the set on which the hierarchy is built. Let us start this section with basic notions on graphs.

We define a graph as a pair $X = (V, E)$ where V is a finite set and E is composed of unordered pairs of distinct elements in V, *i.e.*, E is a subset of $\{\{x, y\} \subseteq \mathrm{V} \mid x \neq y\}$. Each element of V is called a *vertex* of X, and each element of E is called an edge of X.

In the remaining part of this article, we assume that $G = (V, E)$ is a graph and that $\ell = |E|$. Let us now give the definitions of some basic operators on graphs that are used in the sequel.

We denote by $\epsilon^{\times}$ the operator that maps to any subset X of V the subset of E made of the edges of G composed of two vertices in X, *i.e.* $\epsilon^{\times}(X) = \{\{x, y\} \in E \mid x \in X, y \in X\}$. We denote by $\delta^{\bullet}$ the operator that maps any subset F of E to the subset of V made of every vertex of V that belongs to an edge in F, *i.e.*, $\delta^{\bullet}(F) = \cup F$. We denote by $\delta^{\times}$ the operator that maps to any subset X of V the subset of E made of every edge of G that contains a vertex of X, *i.e.*, $\delta^{\times}(X) = \{\{x, y\} \in E \mid \{x, y\} \cap X \neq \emptyset\}$. Finally, we denote by γ the operator that maps to any two subsets X and Y of V the subset of E made of every edge of E that contains exactly one vertex of X and exactly one vertex of Y, *i.e.*, $\gamma(X, Y) = \epsilon^{\times}(X \cup Y) \setminus \epsilon^{\times}(X) \setminus \epsilon^{\times}(Y)$. The set $\gamma(X, Y)$ is called the *common neighborhood of X and Y*

A *total order on E* is a sequence $(u_1, \ldots, u_\ell)$ such that $\{u_1, \ldots, u_\ell\} = E$. Let $\prec$ be a total order on E. Let k in $\{1, \ldots, \ell\}$, we denote by $u_k^{\prec}$ the k-th element of E for the order $\prec$. Let u be an edge in E. The *rank of u for $\prec$*, denoted by $r^{\prec}(u)$, is the unique integer k such that $u = u_k^{\prec}$. Let v be an edge of G. We write $u \prec v$ if the rank of u is less than the one of v ,*i.e.*, $r^{\prec}(u) < r^{\prec}(v)$.

In the remaining part of this article, the symbol $\prec$ denotes a total order on E.

Let $\mathbf{P}$ be a partition of V. We consider the *class map associated with* $\mathbf{P}$ (see [15]), denoted by $\mathrm{Cl}_{\mathbf{P}}$, as the map from V to the set of all subsets of V such that $\mathrm{Cl}_{\mathbf{P}}(x) = \emptyset$ if x does not belong to the support of $\mathbf{P}$ and $\mathrm{Cl}_{\mathbf{P}}(x) = R$ where R is the unique region of $\mathbf{P}$ which contains x, otherwise.

Let $\mathcal{X}$ be a hierarchy on V. Let $\{x, y\}$ be an edge in E and let k be the rank of $\{x, y\}$ for $\prec$. The *update of $\mathcal{X}$ with respect to* $\{x, y\}$, denoted by $\mathcal{X} \oplus \{x, y\}$, is the hierarchy defined by:

$$\begin{aligned} &\mathcal{X} \oplus \{x, y\}[\lambda] = \mathcal{X}[\lambda], \text{ for any } \lambda \text{ in } \{0, \ldots, k-1\}; \text{ and} \\ &\mathcal{X} \oplus \{x, y\}[\lambda] = \mathcal{X}[\lambda] \setminus \{R_x, R_y\} \cup \{R_x \cup R_y\} \\ &\qquad\qquad \text{where } R_x = \mathrm{Cl}_{\mathcal{X}[\lambda]}(x) \text{ and } R_y = \mathrm{Cl}_{\mathcal{X}[\lambda]}(y), \text{ for any } \lambda \text{ in } \{k, \ldots, \ell\}. \end{aligned}$$

Property 3 *Let u and v be two edges in E and let $\mathcal{H}$ be a hierarchy. The following statements hold true:*

$$\mathcal{H} \oplus u = \mathcal{H} \oplus u \oplus u$$
$$\mathcal{H} \oplus u \oplus v = \mathcal{H} \oplus v \oplus u$$

Definition 4 (edge-addition) *Let $E' \subseteq E$ and let $\mathcal{H}$ be a hierarchy. We set $\mathcal{H} \boxplus E' = \mathcal{H} \oplus u_1 \oplus \ldots \oplus u_{|E'|}$ where $E' = \{u_1, \ldots, u_{|E'|}\}$. The binary operation $\boxplus$ is called the* edge-addition*.*

Let X be a set, we denote by $\perp_X$ the hierarchy defined by $\perp_X[\lambda] = \{\{x\} \mid x \in X\}$, for any λ in $\{0, \ldots \ell\}$.

Definition 5 (Binary partition hierarchy) *The* binary partition hierarchy (for $\prec$)*, denoted by $\mathcal{B}^{\prec}$, is the hierarchy $\perp_V \boxplus E$. Let A be a subset of V, we define the* (partial) binary partition hierarchy (for $\prec$) on A*, denoted by $\mathcal{B}^{\prec}_A$, as the hierarchy $\perp_A \boxplus \epsilon^{\times}(A)$.*

Property 6 *The binary partition hierarchy for $\prec$ is complete and binary.*

The notion of a binary partition hierarchy is illustrated in Figs. 1 and 2. In Fig. 1, the hierarchy $\mathcal{H}$ (in bold in the first row) is the binary partition hierarchy of the graph shown in red for the order $\prec$ associated with the ranks given below the edges. In Fig. 2, the two hierarchies $\mathcal{X}$ and $\mathcal{Y}$ represented in the first row in blue and in green are the partial binary partition hierarchies associated to the 8 leftmost (resp. to the 4 rightmost) vertices of the graph.

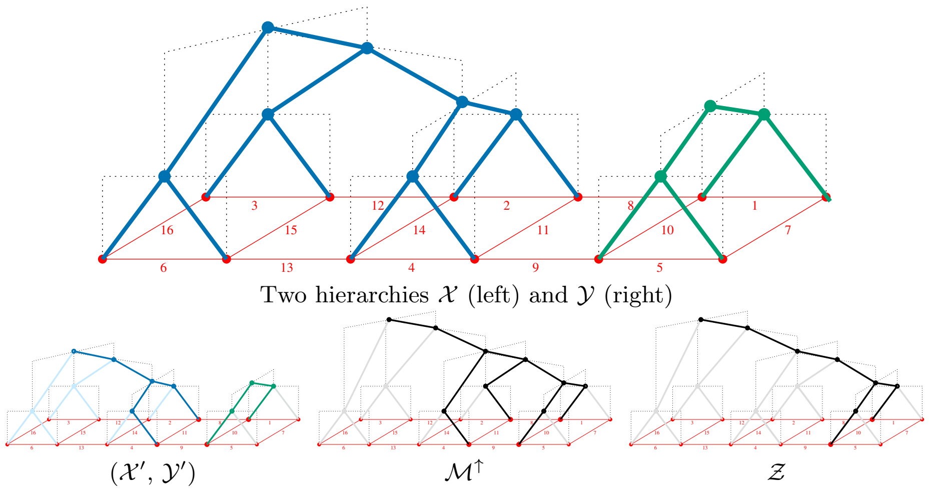
3
2
1
12
8
16
15
14
11
10
7
6
13
4
9
5
Two hierarchies $\mathcal{X}$ (left) and $\mathcal{Y}$ (right)
$(\mathcal{X}', \mathcal{Y}')$
$\mathcal{M}^{\uparrow}$
$\mathcal{Z}$

Fig. 2. Illustration of partial binary partition hierarchies and of the operations select and join. First row: the two partial binary partition hierarchies $\mathcal{X}$ and $\mathcal{Y}$ (represented in blue and in green, respectively) on the sets A and B of the 8 leftmost vertices and of the 4 rightmost vertices, respectively. Second row: (left) two hierarchies $\mathcal{X}'$ and $\mathcal{Y}'$ which are equal to select $(\gamma^{\bullet}_B(A), \mathcal{X})$ and to select $(\gamma^{\bullet}_A(B), \mathcal{Y})$ respectively; (center) the hierarchy $\mathcal{M}^{\uparrow} = \text{join}\,(\mathcal{X}',\ \mathcal{Y}')$; and (right) the hierarchy $\mathcal{Z} = \text{select}\,\left(\gamma^{\bullet}_A(B), \mathcal{M}^{\uparrow}\right)$.

The definition of a binary partition hierarchy given in Definition 5 differs from the one given in [3]. In [3], the edges of E are considered in increasing order of rank, whereas in Definition 5 the edges are added in any order since, from Property 3, the chosen order does not change the resulting hierarchy. From the particular case where the edges are picked in increasing order in Definition 5, it can be observed that the two definitions are equivalent.

In this article, we consider the problem of computing the distribution of the binary partition hierarchy over a partition of the space under the out-of-core constraint. To this end, we finish this section by analyzing some algebraic properties of the external operation $\boxplus$.

Let $\mathcal{X}$ and $\mathcal{Y}$ be two hierarchies in $\mathcal{H}_\ell(V)$. We say that $\mathcal{Y}$ is *smaller than* $\mathcal{X}$, and we write $\mathcal{Y} \sqsubseteq \mathcal{X}$, if, for any λ in $\{0, \ldots, \ell\}$, the partition $\mathcal{Y}[\lambda]$ is a refinement of $\mathcal{X}[\lambda]$. The set $\mathcal{H}_\ell(V)$ equipped with the relation $\sqsubseteq$ is a lattice whose supremum and infimum are denoted by $\sqcup$ and $\sqcap$ respectively. We refer to [16] for a general study of the lattices of (partial) partitions and of hierarchies. Let $u = \{x, y\}$ be an edge in E, we denote by $\mathcal{H}_u$ the hierarchy $\perp_u \oplus u$. We have:

for any λ in $\{0, \ldots, r^{\prec}(u) - 1\}, \mathcal{H}_u[\lambda] = \{\{x\}, \{y\}\}$; and
for any λ in $\{r^{\prec}(u), \ldots, \ell\}, \mathcal{H}_u[\lambda] = \{u\}$.

Property 7 *Let $\mathcal{X}$ and $\mathcal{Y}$ be two hierarchies in $\mathcal{H}_\ell(V)$, let u be an edge in E, and let F and I be two subsets of E. Then, the following equalities hold true:*

1. $\mathcal{H} \oplus u = \mathcal{H} \sqcup \mathcal{H}_u$;
2. $\mathcal{H} \boxplus F = \mathcal{H} \sqcup (\sqcup\{\mathcal{H}_u \mid u \in F\})$;
3. $(\mathcal{X} \sqcup \mathcal{Y}) \boxplus F = \mathcal{X} \sqcup (\mathcal{Y} \boxplus F) = (\mathcal{X} \boxplus F) \sqcup \mathcal{Y}$;
4. $\mathcal{X} \boxplus (F \cup I) = (\mathcal{X} \boxplus F) \boxplus I = (\mathcal{X} \boxplus I) \boxplus F$; *and*
5. $(\mathcal{X} \sqcap \mathcal{Y}) \boxplus F = (\mathcal{X} \boxplus F) \sqcap (\mathcal{Y} \boxplus F)$.

4 Join and insert operations

In this section, we introduce the two main binary operations, called join and insert, that allow us to compute (the distribution of) binary partition hierarchies in an out-of-core manner. Following the high-level schemes presented in [10, 5, 7, 9], the basic idea is to work independently on small pieces of the space, "join" the information found on adjacent pieces and "propagate" (or "insert") this information into other pieces. After presenting the definition of these operations, we investigate some basic properties which are then used in the following section to establish an out-of-core calculus for distributed binary partition hierarchies.

The *extent* of a hierarchy is the union of the set of its regions and its *ground* is the support of its 0-scale. The ground and the extent of a hierarchy $\mathcal{H}$ are denoted by $gr(\mathcal{H})$ and $ex(\mathcal{H})$, respectively.

Definition 8 (join) *Let $\mathcal{X}$ and $\mathcal{Y}$ be two hierarchies in $\mathcal{H}_\ell(V)$. The* join of $\mathcal{X}$ and $\mathcal{Y}$*, denoted by $join(\mathcal{X}, \mathcal{Y})$, is the hierarchy defined by $join(\mathcal{X}, \mathcal{Y}) = (\mathcal{X} \sqcup \mathcal{Y}) \boxplus F$, where F is the common neighborhood of the grounds of $\mathcal{X}$ and of $\mathcal{Y}$,* i.e., *$F = \gamma(gr(\mathcal{X}), gr(\mathcal{Y}))$.*

The operation join is illustrated in Fig. 2. The next property indicates that the binary partition hierarchy for $\prec$ on a set $A \cup B$ can be obtained by first considering the binary partition hierarchies on A and on B and then considering the join of these two hierarchies.

Property 9 *Let A and B be two subsets of V. Then, we have $join(\mathcal{B}^{\prec}_{A},\ \mathcal{B}^{\prec}_{B}) = \mathcal{B}^{\prec}_{A\cup B}$.*

Definition 10 (Insert) *Let $\mathcal{X}$ and $\mathcal{Y}$ be two hierarchies.*

We say that $\mathcal{X}$ is insertable in $\mathcal{Y}$ *if, for any λ in $\{0,\ldots,\ell\}$, for any region Y of $\mathcal{Y}[\lambda]$, Y is either included in a region of $\mathcal{X}[\lambda]$ or is included in $V \setminus gr(\mathcal{X}[\lambda])$.*

Let $\mathcal{X}$ be insertable in $\mathcal{Y}$. The insertion of $\mathcal{X}$ into $\mathcal{Y}$ *is the hierarchy $\mathcal{Z}$, such that, for any λ in $\{0,\ldots,\ell\}$, $\mathcal{Z}[\lambda] = \mathcal{X}[\lambda] \cup \{R \in \mathcal{Y}[\lambda] \mid R \cap gr(\mathcal{X}[\lambda]) = \emptyset\}$. The insertion of $\mathcal{X}$ into $\mathcal{Y}$ is denoted by $insert(\mathcal{X},\mathcal{Y})$.*

For instance, in Fig. 2, the hierarchy $\mathcal{Z}$ is insertable in the hierarchy $\mathcal{Y}$. The insertion of $\mathcal{Z}$ into $\mathcal{Y}$ is the hierarchy depicted in Fig. 1(right-bottom).

Property 11 *Let $\mathcal{X}$ and $\mathcal{Y}$ be two hierarchies. The following statements hold true:*

1. *the hierarchy $\mathcal{X}$ is insertable in $\mathcal{Y}$ if and only if $select(gr(\mathcal{X}),\mathcal{Y}) \sqsubseteq \mathcal{X}$.*
2. *if $\mathcal{X}$ is insertable in $\mathcal{Y}$, then we have $insert(\mathcal{X},\mathcal{Y}) = \mathcal{X} \sqcup \mathcal{Y}$.*

The next lemma states that the selection is distributive over the insertion. Thus, when one needs to calculate a selection of the insertion of a hierarchy into another one, then it is enough to insert a selection of the first hierarchy in the selection of the second. In other words, the insertion can be performed in subparts of the hierarchy instead of on the whole hierarchy, a desirable property in the context of out-of-core computation.

Lemma 12 *Let A be a subset of V and let $\mathcal{X}$ and $\mathcal{Y}$ be two hierarchies such that $\mathcal{X}$ is insertable in $\mathcal{Y}$. Then, we have:*

$$select(A, insert(\mathcal{X},\mathcal{Y})) = insert(select(A,\mathcal{X}), select(A,\mathcal{Y}))$$

The following lemma shows that when one needs to add a certain set of edges to a hierarchy, the addition can be performed "locally" on a subpart of the hierarchy (of the regions containing an element of the edges to be added) before being inserted in the initial hierarchy.

Lemma 13 *Let $\mathcal{X}$ be a hierarchy in $\mathcal{H}_{\ell}(V)$ and let F be a subset of E. Then, we have $\mathcal{X} \boxplus F = insert(select(\delta^{\bullet}(F),\mathcal{X}) \boxplus F, \mathcal{X})$.*

Due to Lemmas 12 and 13, we deduce the following property. It provides us with an out-of-core calculus to obtain the distribution of a binary partition hierarchy over a bi-partition $\{A, B\}$. This calculus is out-of-core in the sense that it does not consider the binary partition hierarchy on the whole set $A \cup B$.

Let A and B be any two subsets of V. We set $\gamma^{\bullet}_{B}(A) = \delta^{\bullet}(\gamma(A,B)) \cap A$. In other words, $\gamma^{\bullet}_{B}(A)$ contains every vertex in A which adjacent to a vertex in B.

Property 14 *Let A and B be two subsets of V.*
Let $\mathcal{M}^{\uparrow} = join\,(select\,(\gamma^{\bullet}_B(A), \mathcal{B}^{\prec}_A),\ select\,(\gamma^{\bullet}_A(B), \mathcal{B}^{\prec}_B))$. Then, we have:

$$select\left(B, \mathcal{B}^{\prec}_{A\cup B}\right) = insert(select\left(\gamma^{\bullet}_A(B), \mathcal{M}^{\uparrow}\right), \mathcal{B}^{\prec}_B).$$

Property 14 is illustrated in Figs. 1 and 2. The first row of Fig. 2 represents two hierarchies $\mathcal{X} = \mathcal{B}^{\prec}_A$ and $\mathcal{Y} = \mathcal{B}^{\prec}_B$, where A (resp B) is the set of the 8 leftmost (resp. 4 rightmost) vertices of the depicted graph. The hierarchies $\mathcal{X}' = \text{select}\,(\gamma^{\bullet}_B(A), \mathcal{B}^{\prec}_A)$, $\mathcal{Y}' = \text{select}\,(\gamma^{\bullet}_A(B), \mathcal{B}^{\prec}_B)$, $\mathcal{M}^{\uparrow} = \text{join}\,(\text{select}\,(\gamma^{\bullet}_B(A), \mathcal{B}^{\prec}_A),\ \text{select}\,(\gamma^{\bullet}_A(B), \mathcal{B}^{\prec}_B))$, and $\mathcal{Z} = \text{select}\left(\gamma^{\bullet}_A(B), \mathcal{M}^{\uparrow}\right)$ are depicted from left to right in the second row of Fig. 2. It can be observed that, as established by Property 14, the insertion of $\mathcal{Z}$ into $\mathcal{X} = \mathcal{B}^{\prec}_A$ is indeed equal to select $(B, \mathcal{B}^{\prec}_{A\cup B})$ (shown in Fig. 1(bottom-right)). The next lemma is the key ingredient to extend the above calculus to partitions with more than two sets.

Lemma 15 *Let A, B and C be three pairwise disjoint subsets of V such that $\gamma(A, C) = \emptyset$. Let $\mathcal{M}^{\uparrow} = join\,(select(\gamma^{\bullet}_B(A), \mathcal{B}^{\prec}_A),\ select(\gamma^{\bullet}_A(B), \mathcal{B}^{\prec}_B))$ and let $\mathcal{M}^{\downarrow} = join\,(select\,(\gamma^{\bullet}_B(A), \mathcal{B}^{\prec}_A),\ select\,(\gamma^{\bullet}_A(B), \mathcal{B}^{\prec}_{B\cup C}))$. Then, we have :*

1. *$select\,(A, \mathcal{B}^{\prec}_{A\cup B\cup C}) = insert(select\left(\gamma^{\bullet}_B(A), \mathcal{M}^{\downarrow}\right), \mathcal{B}^{\prec}_A)$; and*
2. *$\mathcal{M}^{\downarrow} = insert(select\,(\gamma^{\bullet}_A(B), \mathcal{B}^{\prec}_{A\cup B\cup C}), \mathcal{M}^{\uparrow})$.*

5 Out-of-core calculus

In this section, after providing the definition of a causal partition, we introduce an out-of-core calculus (with an associated algorithm) to efficiently obtain the distribution of a binary partition hierarchy on a causal partition. The main result of this section (Theorem 17) states the correctness of this calculus.

A causal partition of V is a series $(S_0, \ldots, S_k)$ of subsets of V such that: (i) $\{S_0, \ldots, S_k\}$ is a partition of V; (ii) $\delta^{\times}(S_0) = \epsilon^{\times}(S_0) \cup \gamma(S_0, S_1)$; (iii) $\delta^{\times}(S_i) = \epsilon^{\times}(S_i) \cup \gamma(S_i, S_{i-1}) \cup \gamma(S_i, S_{i+1})$, for any i in $\{1, \ldots, k-1\}$; and (iv) $\delta^{\times}(S_k) = \epsilon^{\times}(S_k) \cup \gamma(S_k, S_{k-1})$. Any element of a causal partition δ is called a *slice of δ*.

Let us now present induction formulae to compute the distribution of a binary partition hierarchy over a causal partition of the space.

Definition 16 *Let $(S_0, \ldots, S_k)$ be a causal partition of V. We set*

1. *$\mathcal{B}^{\uparrow}_0 = \mathcal{B}^{\prec}_{S_0}$;*
2. *$\forall i \in \{1, \ldots, k\}$, $\mathcal{M}^{\uparrow}_i = join\left(select\left(\gamma^{\bullet}_{S_i}(S_{i-1}), \mathcal{B}^{\uparrow}_{i-1}\right),\ select\left(\gamma^{\bullet}_{S_{i-1}}(S_i), \mathcal{B}^{\prec}_{S_i}\right)\right)$;*
3. *$\forall i \in \{1, \ldots, k\}$, $\mathcal{B}^{\uparrow}_i = insert(select\left(\gamma^{\bullet}_{S_{i-1}}(S_i), \mathcal{M}^{\uparrow}_i\right), \mathcal{B}^{\prec}_{S_i})$;*
4. *$\mathcal{B}^{\downarrow}_k = \mathcal{B}^{\uparrow}_k$ and $\mathcal{M}^{\downarrow}_k = \mathcal{M}^{\uparrow}_k$;*
5. *$\forall i \in \{0, \ldots, k-1\}$, $\mathcal{B}^{\downarrow}_i = insert(select\left(\gamma^{\bullet}_{S_{i-1}}(S_i), \mathcal{M}^{\downarrow}_{i+1}\right), \mathcal{B}^{\uparrow}_i)$; and*
6. *$\forall i \in \{1, \ldots, k-1\}$, $\mathcal{M}^{\downarrow}_i = insert(select\left(\gamma^{\bullet}_{S_{i-1}}(S_i), \mathcal{B}^{\downarrow}_i\right), \mathcal{M}^{\uparrow}_i)$.*

On the one hand, the three first formulae of Definition 16 provide us with a causal (inductive) calculus: i-th term of the series relies on the $(i-1)$-th term of the series. In particular, $\mathcal{M}_i^{\uparrow}$ depends on $\mathcal{B}_{i-1}^{\uparrow}$ and that $\mathcal{B}_i^{\uparrow}$ depends on $\mathcal{M}_i^{\uparrow}$, for any i in $\{1, \ldots, k\}$. On the other hand, the three last formulae of Definition 16 provide us with an anti-causal calculus since the terms of index i rely on the ones of index $(i+1)$: $\mathcal{B}_i^{\downarrow}$ depends on $\mathcal{M}_{i+1}^{\downarrow}$ and $\mathcal{M}_i^{\downarrow}$ depends on $\mathcal{B}_i^{\downarrow}$. Note also that $\mathcal{M}_i^{\downarrow}$ depends on $\mathcal{M}_i^{\uparrow}$ and that $\mathcal{B}_i^{\downarrow}$ depends on $\mathcal{B}_i^{\uparrow}$. Hence, the anti-causal calculus must be performed after the causal ones. One can also observe that the computation of each term requires only the knowledge of hierarchies whose grounds are included in two consecutive slices of the given causal partition. In this sense, we can say that the above formulae form an out-of-core calculus. Algorithm 1, presented hereafter, is a sequential implementation of the above induction formulae. Given a partition of V into $k+1$ slices, it can be observed that Algorithm 1 performs $O(k)$ calls to select, insert, join, and to the algorithm PlayingWithKruskal [12] applied to a single slice of the given partition.

Algorithm 1: Out-of-core binary partition hierarchy.

```
Data: A graph $(V, E)$, a total order $\prec$ on $E$, and a causal partition
      $(S_0, \ldots, S_k)$ of $V$
Result: $\{\mathcal{B}_0^{\downarrow}, \ldots, \mathcal{B}_k^{\downarrow}\}$: the distribution of the binary partition hierarchy $\mathcal{B}_V^{\prec}$
      over $\{S_0, \ldots, S_k\}$.
1 $\mathcal{B}_0^{\uparrow} := \mathcal{B}_{S_0}^{\prec}$          // Def. 16.1, call PlayingWithKruskal algorithm
2 foreach $i$ from 1 to $k$ do                                   // Causal traversal of the slices
3     Call PlayingWithKruskal algorithm to compute $\mathcal{B}_{S_i}^{\prec}$
4     $\mathcal{M}_i^{\uparrow} :=$ join$\left(\text{select}\left(\gamma^{\bullet}_{S_i}(S_{i-1}), \mathcal{B}_{i-1}^{\uparrow}\right), \text{select}\left(\gamma^{\bullet}_{S_{i-1}}(S_i), \mathcal{B}_{S_i}^{\prec}\right)\right)$ // Def. 16.2
5     $\mathcal{B}_i^{\uparrow} :=$ insert(select$\left(\gamma^{\bullet}_{S_{i-1}}(S_i), \mathcal{M}_i^{\uparrow}\right), \mathcal{B}_i^{\prec})$    // Def. 16.3
6 $\mathcal{B}_k^{\downarrow} := \mathcal{B}_k^{\uparrow}$; $\mathcal{M}_k^{\downarrow} := \mathcal{M}_k^{\uparrow}$          // Def. 16.4
7 foreach $i$ from $k-1$ to 0 do                                 // Anticausal traversal of the slices
8     $\mathcal{B}_i^{\downarrow} :=$ insert(select$\left(\gamma^{\bullet}_{S_{i+1}}(S_i), \mathcal{M}_{i+1}^{\downarrow}\right), \mathcal{B}_i^{\uparrow})$    // Def. 16.5
9     if $i > 0$ then $\mathcal{M}_i^{\downarrow} :=$ insert(select$\left(\gamma^{\bullet}_{S_{i-1}}(S_i), \mathcal{B}_i^{\downarrow}\right), \mathcal{M}_i^{\uparrow})$    // Def. 16.6
```

Let us finish this section by Theorem 17 that establishes that the formulae of Definition 16 indeed allows one to compute a distributed binary partition hierarchy, hence, establishing the correctness of Algorithm 1.

Theorem 17 *Let $(S_0, \ldots, S_k)$ be a causal partition of V. Let i be any element in $\{0, \ldots, k\}$. The following statements hold true:*

1. $\mathcal{B}_i^{\uparrow} = select\left(S_i, \mathcal{B}^{\prec}_{\cup\{S_j \mid j \in \{0, \ldots i\}\}}\right)$;
2. $\mathcal{B}_i^{\downarrow} = select(S_i, \mathcal{B}_V^{\prec})$; *and*
3. *the set* $\left\{\mathcal{B}_i^{\downarrow} \mid i \in \{0, \ldots, k\}\right\}$ *is the distribution of* $\mathcal{B}_V^{\prec}$ *over* $\{S_0, \ldots, S_k\}$.

6 Discussion and conclusion

In this article, an algebraic framework to study binary partition hierarchies is presented. Based on this framework, a new calculus allowing us to obtain the binary partition hierarchy associated to a given total order of the edges of a graph is introduced: the result of this calculus is a distributed representation of the binary partition hierarchy where the hierarchy is “split into subparts”, each of them containing the hierarchical information associated to one predefined subset, called a slice, of the space. The proposed calculus only comprises binary operations that depend on two partial hierarchies. Thus, if the size of the partial hierarchies is limited compared to the size of the full hierarchy, computing the result of each operation only requires loading data structures of limited size into memory. In this sense, the proposed calculus is out-of-core. In order to test the practical efficiency of this calculus, an implementation in Python is designed and made available online at https://perso.esiee.fr/~coustyj/hoocc/. Fig. 3 shows the first results obtained for images of 1K $\times$ 1K, 2K $\times$ 2K, and of 4K $\times$ 4K pixels: on the left, the average sizes of the partial hierarchies are displayed as functions of the number of slices into which the image domain is partitioned and, on the right, the execution times per pixel are displayed as functions of the number of slices. It can be seen that the sizes of the partial hierarchies decrease as an inverse function of the number of slices whereas the execution times increase linearly with respect to the number of slices as expected from the theoretical analysis. For instance, compared to the standard case where the image is considered as a single slice, for an image of 16M pixels, the average size of the partial structures is divided by approximately 10, when the space is partitioned into 16 slices while the execution time is multiplied by approximately 3. A detailed description of the algorithms for computing the result of the operations select, join, and insert will be provided in forthcoming articles. Directions for future work also include extensions of the calculus to partitions of the space which are not necessarily causal (*e.g.* grid-like partitions), out-of-core algorithms for image processing based on binary partition hierarchies like (hierarchical) watersheds or marker-based segmentation, and the adaptation of the proposed algorithm to parallel computations.

References

1. Chierchia, G., Perret, B.: Ultrametric fitting by gradient descent. In: NeurIPS. pp. 3181–3192 (2019)
2. Cousty, J., Najman, L., Kenmochi, Y., Guimarães, S.: Hierarchical segmentations with graphs: quasi-flat zones, minimum spanning trees, and saliency maps. JMIV **60**(4), 479–502 (2018)
3. Cousty, J., Najman, L., Perret, B.: Constructive links between some morphological hierarchies on edge-weighted graphs. In: ISMM. pp. 86–97 (2013)
4. Farabet, C., Couprie, C., Najman, L., LeCun, Y.: Learning hierarchical features for scene labeling. TPAMI **35**(8), 1915–1929 (2012)
5. Gazagnes, S., Wilkinson, M.H.: Distributed component forests in 2-d: Hierarchical image representations suitable for tera-scale images. IJPRAI **33**(11), 1940012 (2019)

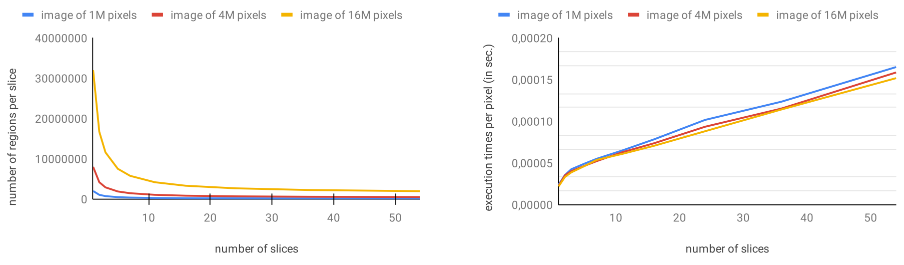

Fig. 3. Left: average size of the hierarchies in distributions of binary partition hierarchies over causal partitions; and right: execution times of the proposed calculus from three images of 1000 × 1000, 2000 × 2000, and of 4000 × 4000 pixels.

6. Gigli, L., Velasco-Forero, S., Marcotegui, B.: On minimum spanning tree streaming for hierarchical segmentation. PRL **138**, 155–162 (2020)
7. Götz, M., Cavallaro, G., Geraud, T., Book, M., Riedel, M.: Parallel computation of component trees on distributed memory machines. TPDS **29**(11), 2582–2598 (2018)
8. Guimarães, S., Kenmochi, Y., Cousty, J., Patrocinio, Z., Najman, L.: Hierarchizing graph-based image segmentation algorithms relying on region dissimilarity: the case of the Felzenszwalb-Huttenlocher method. MMTA **2**(1), 55–75 (2017)
9. Havel, J., Merciol, F., Lefèvre, S.: Efficient tree construction for multiscale image representation and processing. JRTIP **16**(4), 1129–1146 (2019)
10. Kazemier, J.J., Ouzounis, G.K., Wilkinson, M.H.: Connected morphological attribute filters on distributed memory parallel machines. In: ISMM. pp. 357–368 (2017)
11. Maninis, K.K., Pont-Tuset, J., Arbeláez, P., Van Gool, L.: Convolutional oriented boundaries: From image segmentation to high-level tasks. TPAMI **40**(4), 819–833 (2017)
12. Najman, L., Cousty, J., Perret, B.: Playing with kruskal: algorithms for morphological trees in edge-weighted graphs. In: ISMM. pp. 135–146 (2013)
13. Najman, L., Schmitt, M.: Geodesic saliency of watershed contours and hierarchical segmentation. TPAMI **18**(12), 1163–1173 (1996)
14. Perret, B., Cousty, J., Guimarães, S.J.F., Kenmochi, Y., Najman, L.: Removing non-significant regions in hierarchical clustering and segmentation. PRL **128**, 433–439 (2019)
15. Ronse, C.: Partial partitions, partial connections and connective segmentation. JMIV **32**(2), 97–125 (2008)
16. Ronse, C.: Ordering partial partitions for image segmentation and filtering: Merging, creating and inflating blocks. JMIV **49**(1), 202–233 (2014)
17. Salembier, P., Garrido, L.: Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. TIP **9**(4), 561–576 (2000)
18. Soille, P.: Constrained connectivity for hierarchical image partitioning and simplification. TPAMI **30**(7), 1132–1145 (2008)