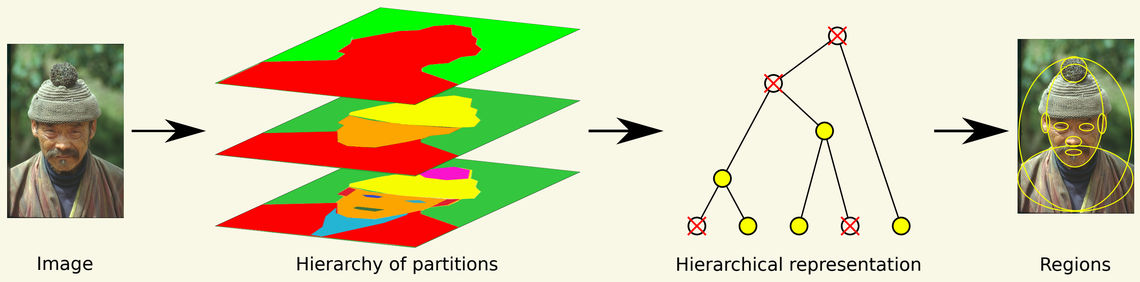
Select, insert and join: out-of-core efficient algorithms for hierarchies of segmentation

Josselin Lefèvre[1,2], Jean Cousty[1], Benjamin Perret[1], Harold Phelippeau[2]

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

Motivation

Image segmentation is one of the oldest problems in computer vision. However, objects of interest do not all appear at the same scale and may be nested within each other, making segmentation an ill-posed problem.



- A hierarchy is a series of nested partitions of a (image) domain:
  - a series $(\mathbf{P}_0, \ldots, \mathbf{P}_\ell)$ of partitions of a set V such that for any i in $\{0, \ldots, \ell - 1\}$, each element of $\mathbf{P}_i$ is included in an element of $\mathbf{P}_{i+1}$.

The Binary Partition Hierarchy and the Minimum Spanning Tree are key structures for hierarchical analysis: (hierarchical) watersheds, constrained connectivity, quasi-flat zones, ultrametric opening, etc.

Problem

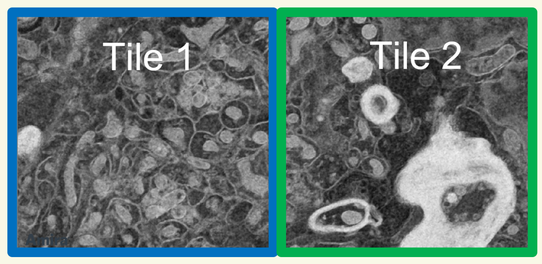
- When the image exceeds a certain size:
  - the data cannot fit in the main memory;
  - the usual sequential algorithm fails to produce a result.
- Examples: biological, astronomical, satellite images, etc.

Solution: compute an out-of-core structure

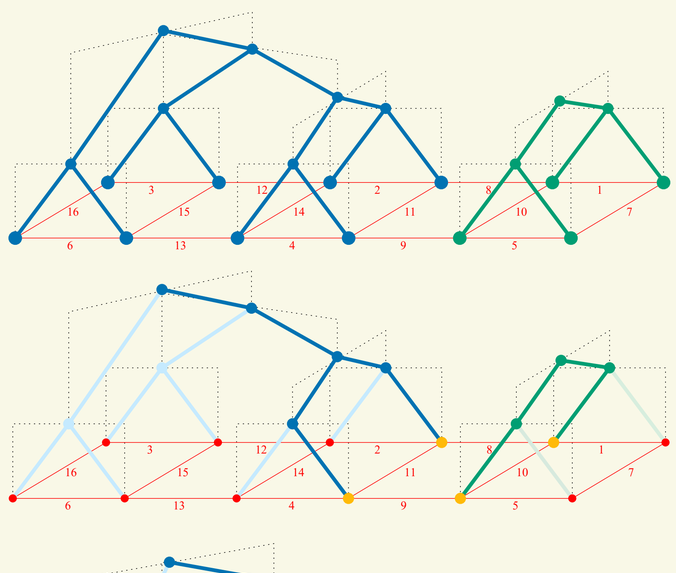
- Produce the same result as the usual algorithms.
- While minimizing the size of the data structures that are simultaneously needed at the different computation steps.

Select, join and insert - causal pass

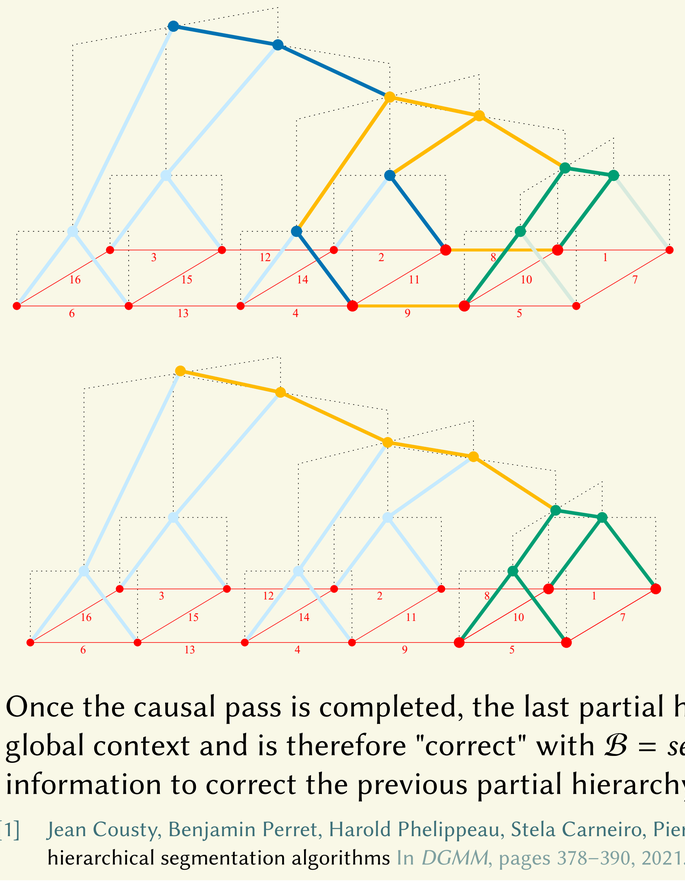
In order to calculate the distribution of a binary partition hierarchy, we will use the three operations introduced in [1] :



Our method is applied on a *causal partition of* V, that is, for $k \in \mathbb{N}$, the sequence $(S_0, \ldots, S_k)$ such that for each t in $0, \cdots, k\}$, $S_t = \{(i,j) \in V \mid t \times \frac{w}{k} \leq i < (t+1) \times \frac{w}{k}$. Each element of this partition is called a **tile**. Let us consider a bi-partition of the space (S_0, S_1).



First, for each region of the partition (S_0, S_1), we compute the binary partition hierarchys locally to obtain the **partial hierarchies** $\mathcal{X}$ and $\mathcal{Y}$.



For each hierarchy, extract the **border trees**. That is to say, for $\mathcal{X}$, the set of regions containing an element of S_0 neighbor to one of $S1$ noted $\gamma^{\bullet}_{S_1}(S_0)$ (respectively $\gamma^{\bullet}_{S_0}(S_1)$ for $\mathcal{Y}$). We thus obtain the edge trees $\mathcal{X}' = \mathit{select}\,(S_0, S_1)$ and $\mathcal{Y}' = \mathit{select}\,(S_1, S_0)$.

We can then join the information of the two edge trees thanks to the **common neighborhood** of the supports of $\mathcal{X}$ and $\mathcal{Y}$, the set of edges having one end in S_0 and the other in S_1, in order to obtain $\mathcal{M} = \mathit{join}\,(\mathcal{X}',\ \mathcal{Y}')$.

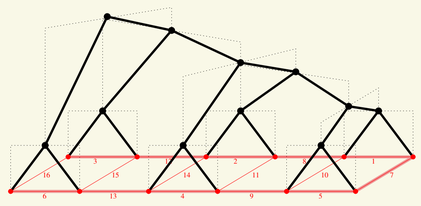
In order to correct $\mathcal{Y}$, we need to insert the information of $\mathcal{M}$ in it through $\mathcal{M}' = \mathit{select}\left(\gamma^{\bullet}_{S_0}(S_1), \mathcal{M}\right)$. We then obtain the **local hierarchy** enriched by the global information $\mathcal{B} = \mathit{insert}(\mathcal{M}', \mathcal{Y})$.

Once the causal pass is completed, the last partial hierarchy considered, $\mathcal{Y}$, has benefited from the entire global context and is therefore "correct" with $\mathcal{B} = \mathit{select}\,(\mathcal{H}, S_1)$. It only remains to retro propagate the information to correct the previous partial hierarchy, $\mathcal{X}$.

[1] Jean Cousty, Benjamin Perret, Harold Phelippeau, Stela Carneiro, Pierre Kamlay, and Lilian Buzer. An algebraic framework for out-of-core hierarchical segmentation algorithms In *DGMM*, pages 378–390, 2021.

Binary partition hierarchy and minimum spanning tree

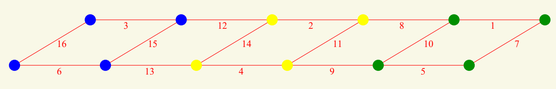
The binary partition hierarchy is the hierarchy obtained during the execution of the Kruskal algorithm allowing to obtain a minimum spanning tree A of a given $G = (V, E, w)$. Each leaf is associated to a vertex of A and each internal node is an edge of A.



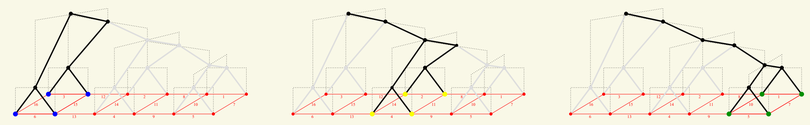
In black a binary partition tree and in bold red its associated minimum spanning tree.

Formal problem statement

Given:



Find:



- Compute the distribution of the binary partition hierarchy $\mathcal{H}$ of $(G, \prec)$ over a causal partition of its ground set.
  - Without compute $\mathcal{H}$.
  - Where each computation step requires a limited amount of memory.

A high level calculus

Data: A graph (V, E), a total order $\prec$ over E, and a causal partition $(S_0, \ldots, S_k)$ of V.

Result: $\{\mathcal{B}^{\downarrow}_0, \ldots, \mathcal{B}^{\downarrow}_k\}$: a distribution of the binary partition hierarchy $\mathcal{B}^{\prec}_V$ over $\{S_0, \ldots, S_k\}$.

$\mathcal{B}^{\uparrow}_0 := \mathcal{B}^{\prec}_{S_0}$ `// Call the algorithm PlayingWithKruskal [2]`

foreach *i from* 1 *to* k **do** `// Causal pass`

  Call PlayingWithKruskal to compute $\mathcal{B}^{\prec}_{S_i}$

  $\mathcal{M}^{\uparrow}_i := \mathit{join}\left(\mathit{select}\left(\gamma^{\bullet}_{S_i}(S_{i-1}), \mathcal{B}^{\uparrow}_{i-1}\right),\ \mathit{select}\left(\gamma^{\bullet}_{S_{i-1}}(S_i), \mathcal{B}^{\prec}_{S_i}\right)\right)$

  $\mathcal{B}^{\uparrow}_i := \mathit{insert}(\mathit{select}\left(\gamma^{\bullet}_{S_{i-1}}(S_i), \mathcal{M}^{\uparrow}_i\right), \mathcal{B}^{\prec}_{S_i})$

end

$\mathcal{B}^{\downarrow}_k := \mathcal{B}^{\uparrow}_k$; $\mathcal{M}^{\downarrow}_k := \mathcal{M}^{\uparrow}_k$

foreach *i from* $k-1$ *to* 0 **do** `// Anticausal pass`

  $\mathcal{B}^{\downarrow}_i := \mathit{insert}(\mathit{select}\left(\gamma^{\bullet}_{S_{i+1}}(S_i), \mathcal{M}^{\downarrow}_{i+1}\right), \mathcal{B}^{\uparrow}_i)$

  if $i > 0$ **then** $\mathcal{M}^{\downarrow}_i := \mathit{insert}(\mathit{select}\left(\gamma^{\bullet}_{S_{i-1}}(S_i), \mathcal{B}^{\downarrow}_i\right), \mathcal{M}^{\uparrow}_i)$

end

- $O(k)$ operations select, join and insert, where k is the number of regions in which the data is split.

[2] Laurent Najman, Jean Cousty, and Benjamin Perret. Playing with Kruskal: algorithms for morphological trees in edge-weighted graphs. In *ISMM*, pages 135–146, 2013.

Efficient implementations

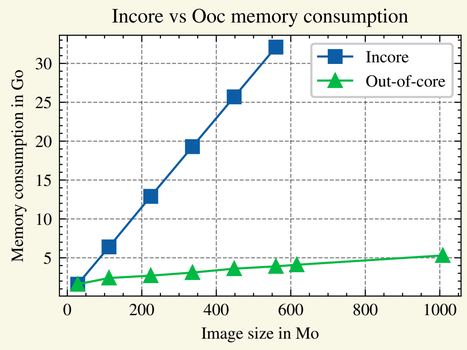
The high-level calculation is based on three operations for which we have given efficient implementations [3].

- **Select** and **Insert** have linear complexity with respect to the number of regions in the input hierarchies.
- The complexity of **Join** is log-linear because it is dominated by a sorting on the edges of the common neighborhood.

[3] Josselin Lefèvre, Jean Cousty, Benjamin Perret, and Harold Phelippeau. Join, select, and insert: Efficient out-of-core algorithms for hierarchical segmentation trees. In *DGMM*, pages 274–286, 2022.

Memory consumption

- Experimentation with a constant tile size on a workstation with 32Go RAM.
- The in-core algorithm does not allow not compute a binary partition hierarchy with an image size over 560 MB.
- Execution time for 10 tiles (200 Mega pixels): 38.3s for in-core computation and 83.8s for the out-of-core one (including 20% of disk access).



Future works

Enrich this out-of-core framework to compute attributes, connected filters, compute (hierarchical) watersheds and state of the art interactive segmentation framework.

josselin.lefevre@thermofisher.com - 2023