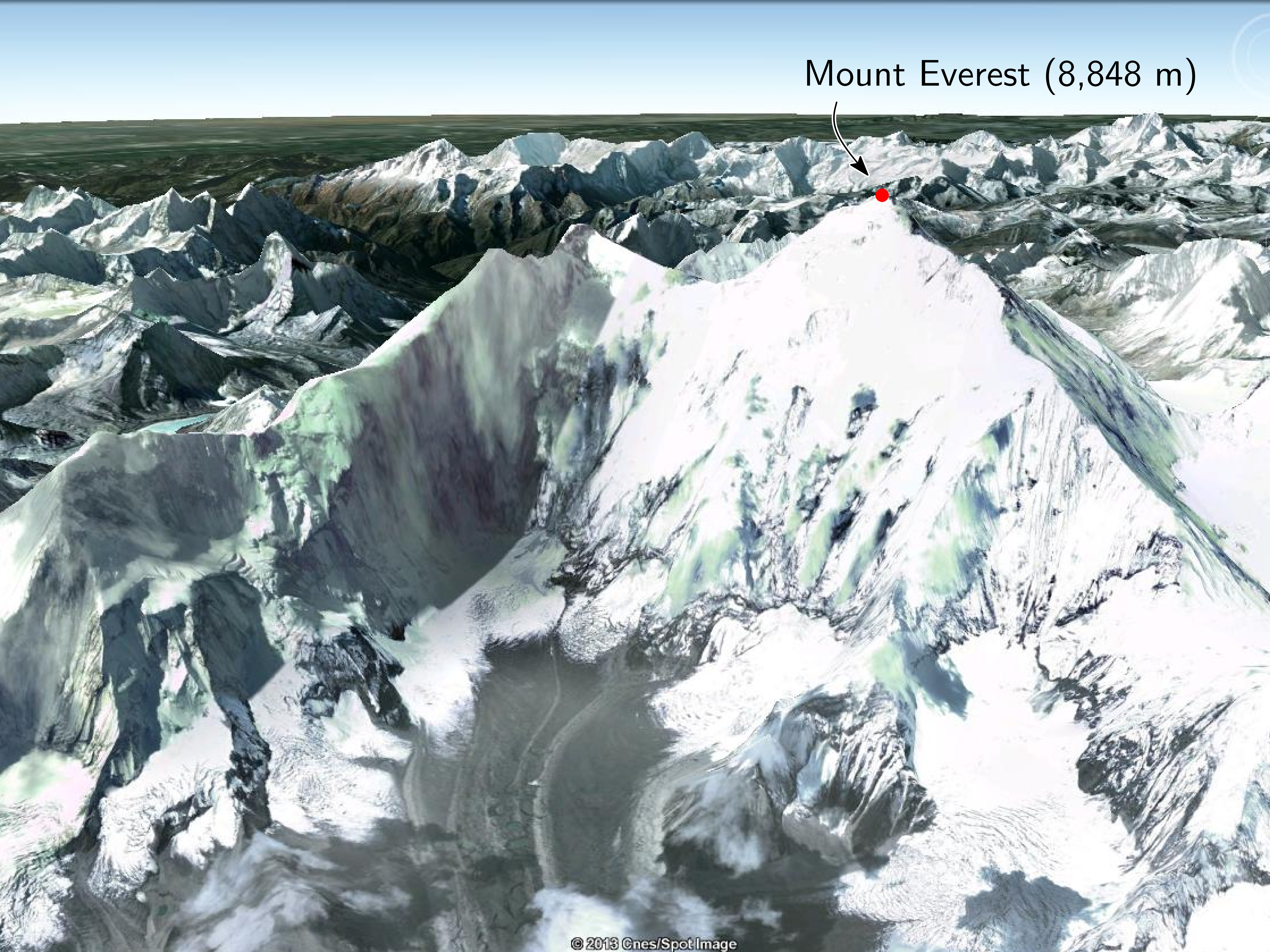
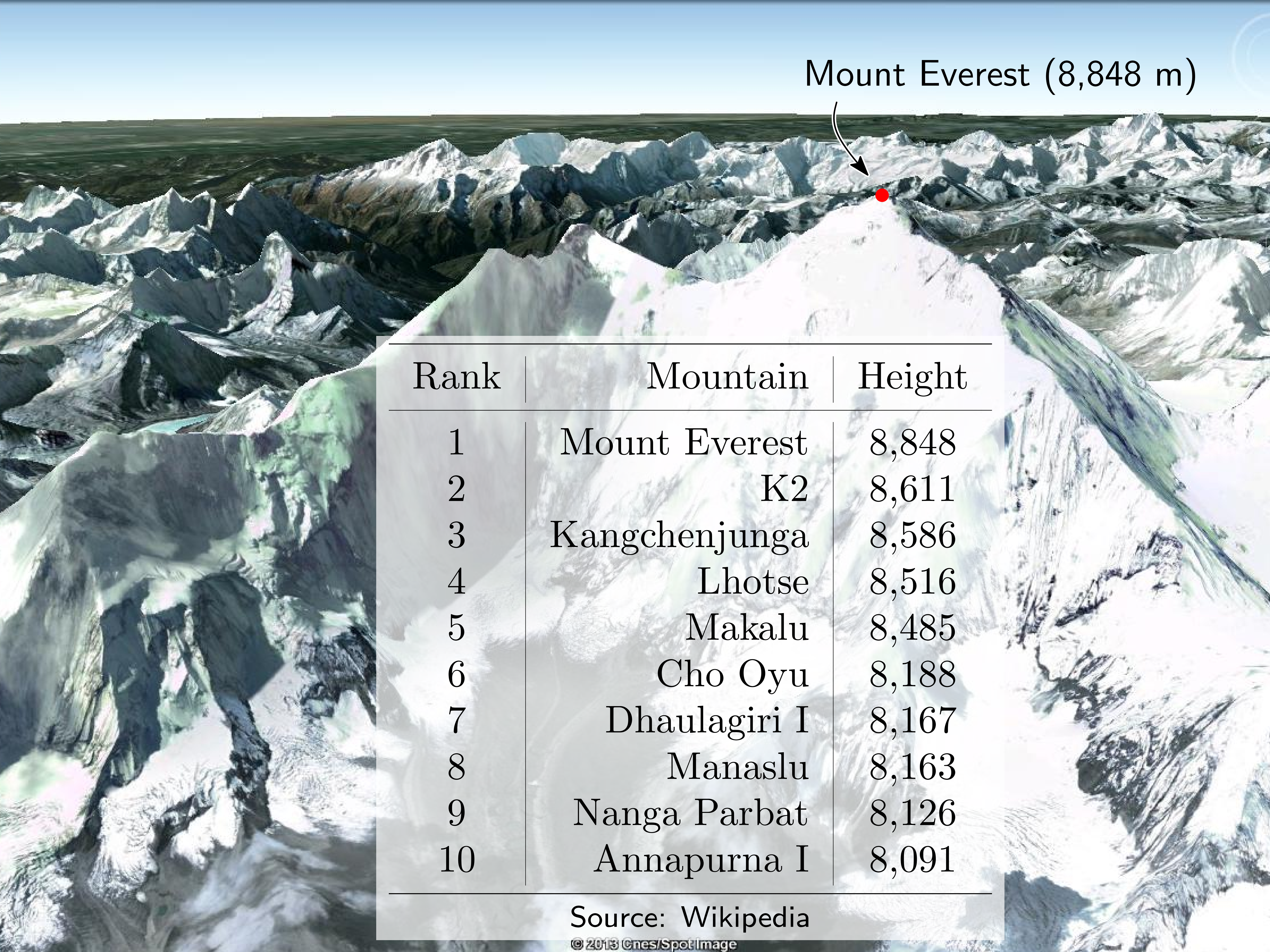
Back to Basics: Merge Trees

Dmitriy Morozov
Lawrence Berkeley National Laboratory

Based on joint works with Kenes Beketayev and Gunther Weber.

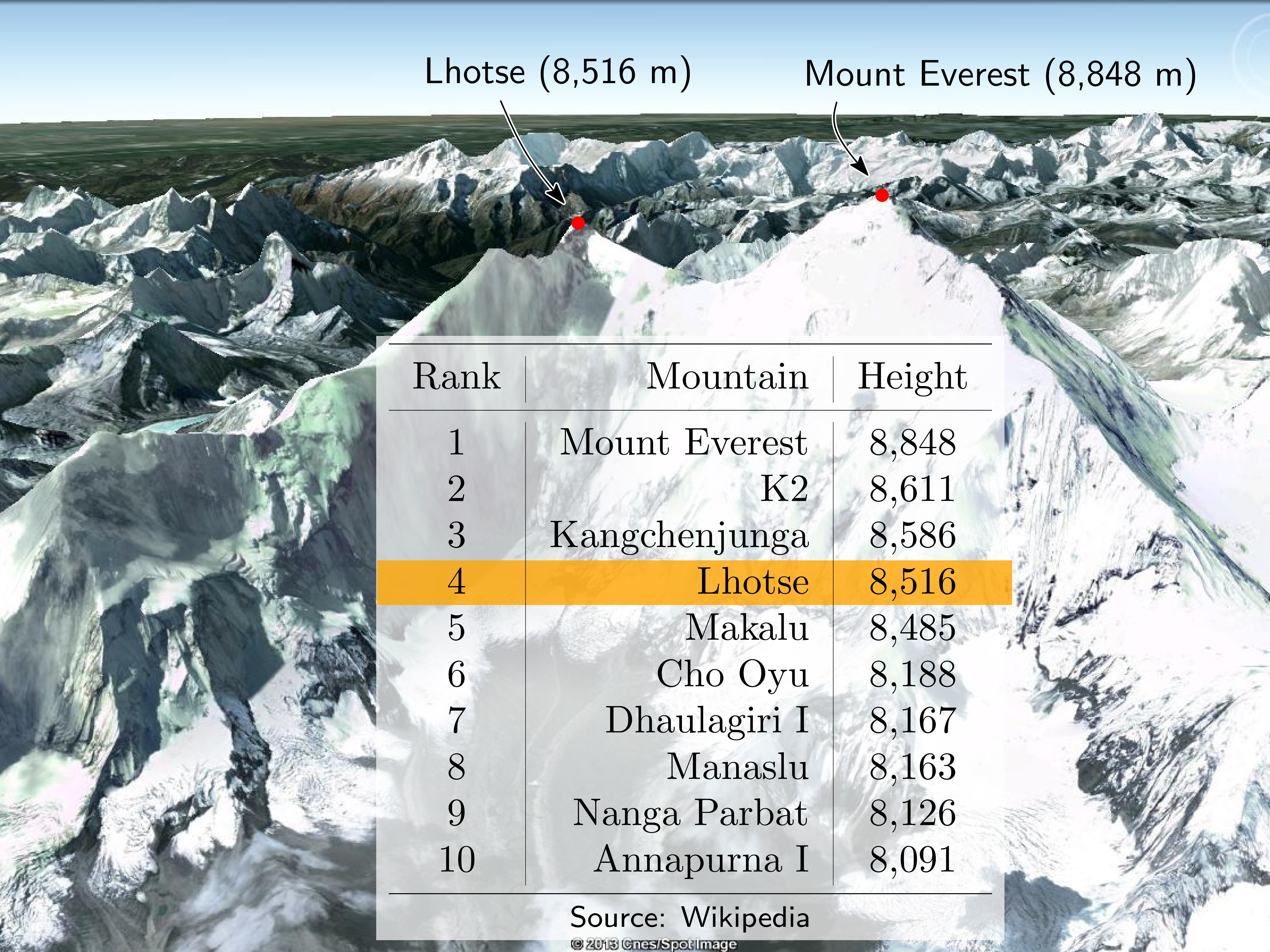
Applied and Computational Algebraic Topology
Bremen, Germany
July 15, 2013


Mount Everest (8,848 m)
© 2013 Cnes/Spot Image



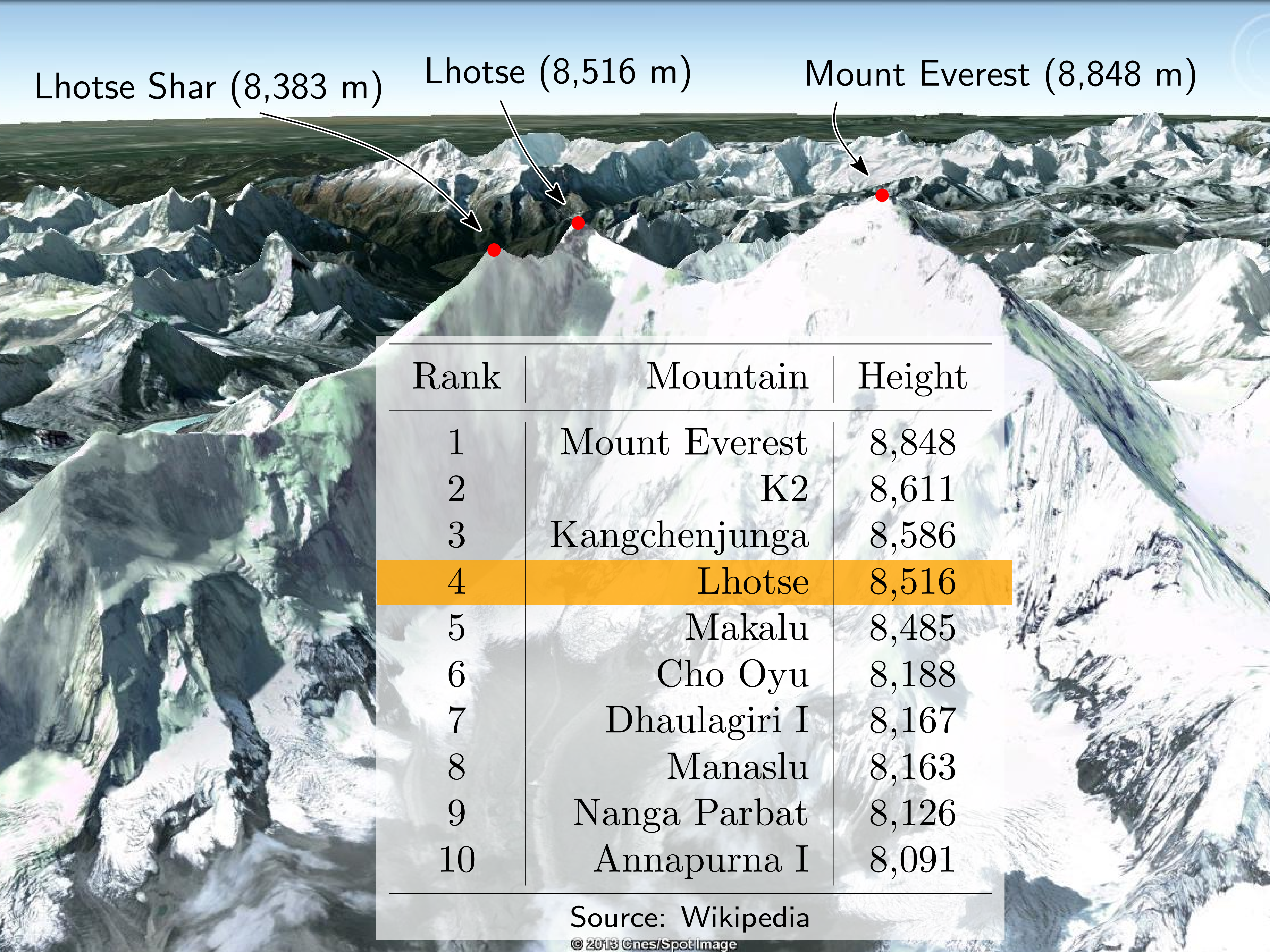
Rank	Mountain	Height
1	Mount Everest	8,848
2	K2	8,611
3	Kangchenjunga	8,586
4	Lhotse	8,516
5	Makalu	8,485
6	Cho Oyu	8,188
7	Dhaulagiri I	8,167
8	Manaslu	8,163
9	Nanga Parbat	8,126
10	Annapurna I	8,091

Source: Wikipedia



Rank	Mountain	Height
1	Mount Everest	8,848
2	K2	8,611
3	Kangchenjunga	8,586
4	Lhotse	8,516
5	Makalu	8,485
6	Cho Oyu	8,188
7	Dhaulagiri I	8,167
8	Manaslu	8,163
9	Nanga Parbat	8,126
10	Annapurna I	8,091

Source: Wikipedia

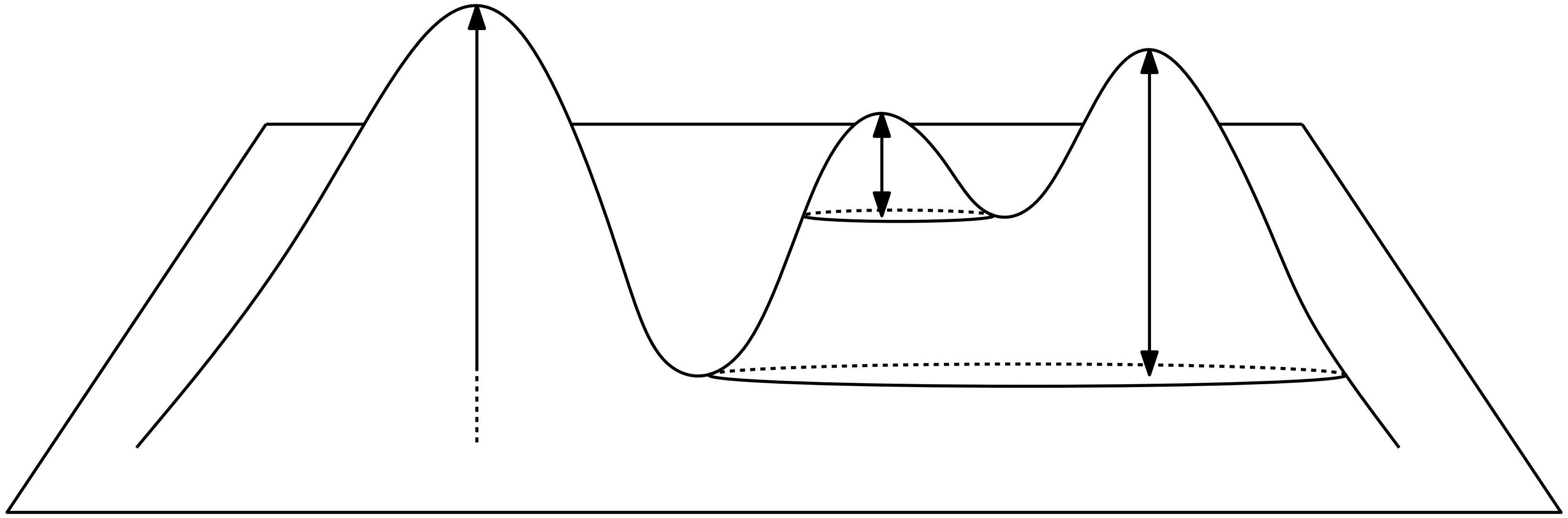


Rank	Mountain	Height
1	Mount Everest	8,848
2	K2	8,611
3	Kangchenjunga	8,586
4	Lhotse	8,516
5	Makalu	8,485
6	Cho Oyu	8,188
7	Dhaulagiri I	8,167
8	Manaslu	8,163
9	Nanga Parbat	8,126
10	Annapurna I	8,091

Source: Wikipedia

Topographic Prominence

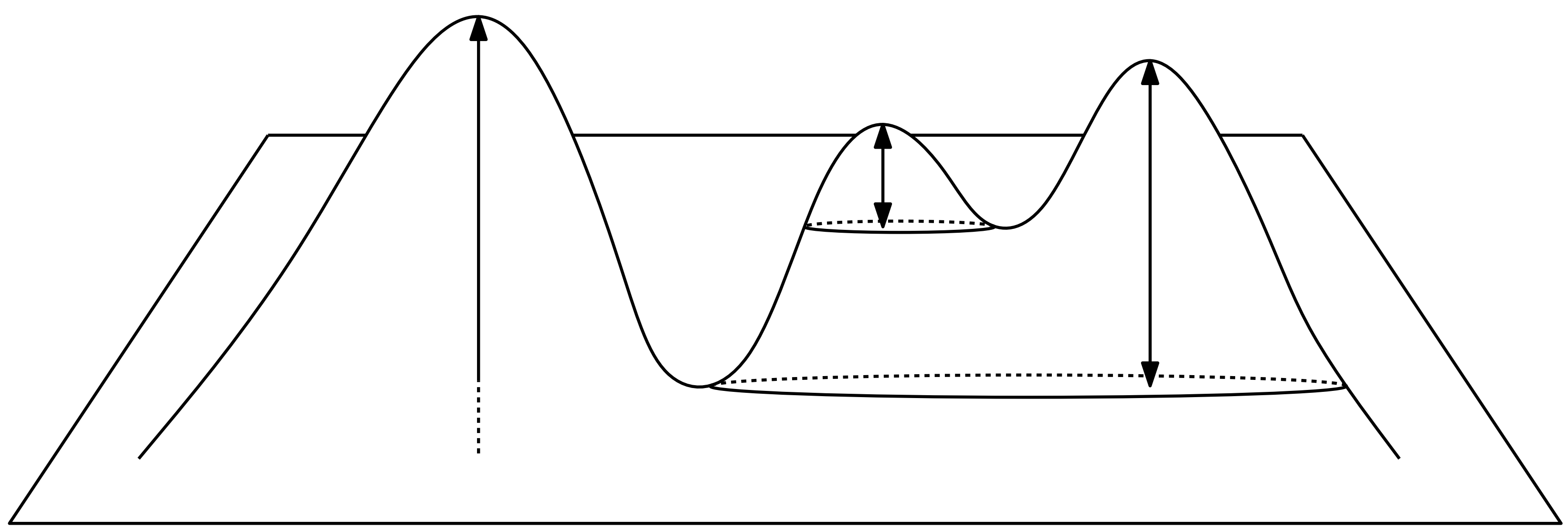
The **prominence** of a peak is the height of the peak's summit above the lowest contour line encircling it and no higher summit.

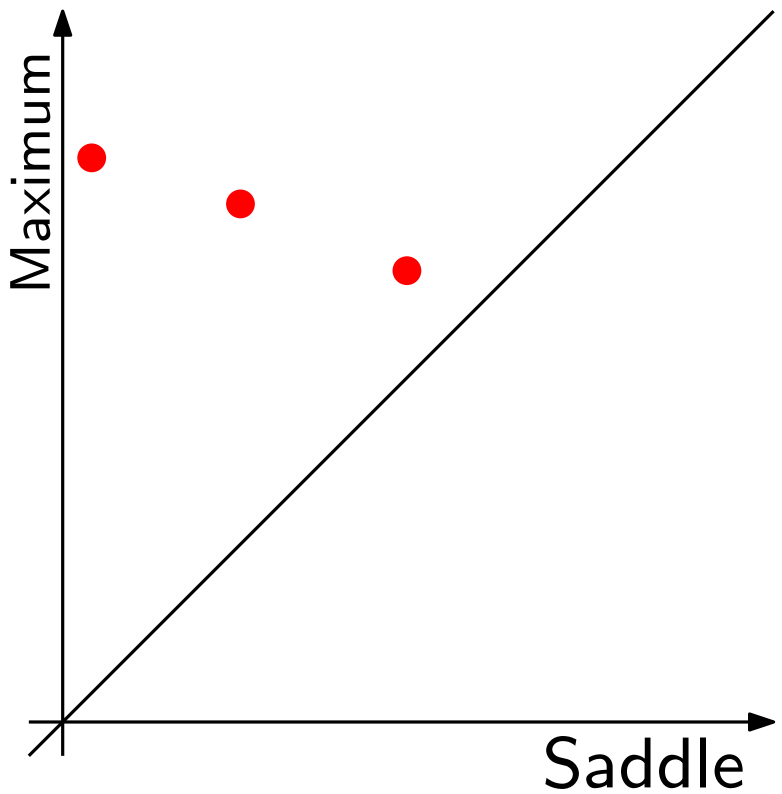


Topographic Prominence

The **prominence** of a peak is the height of the peak's summit above the lowest contour line encircling it and no higher summit.

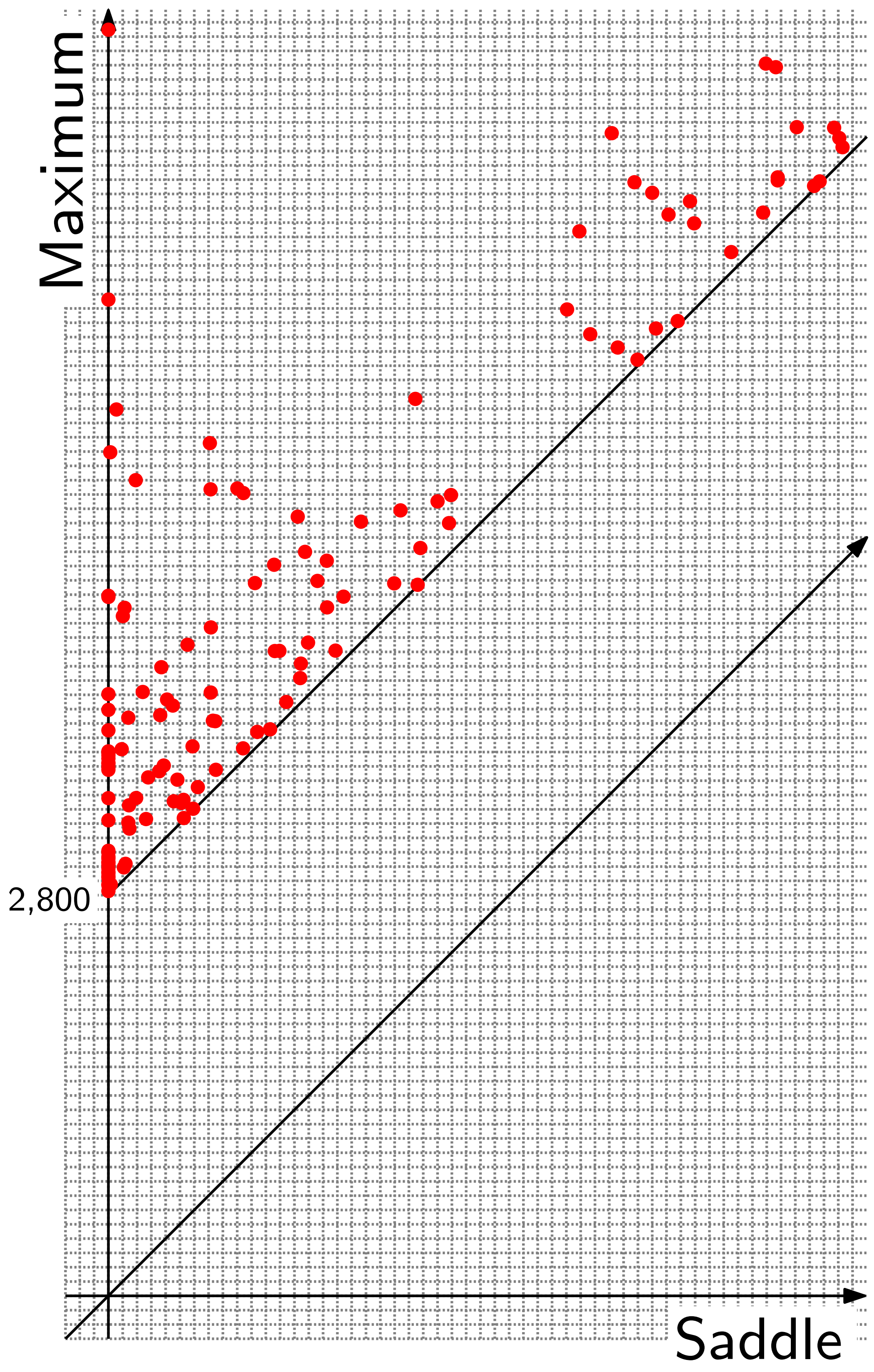
prominence = **persistence**



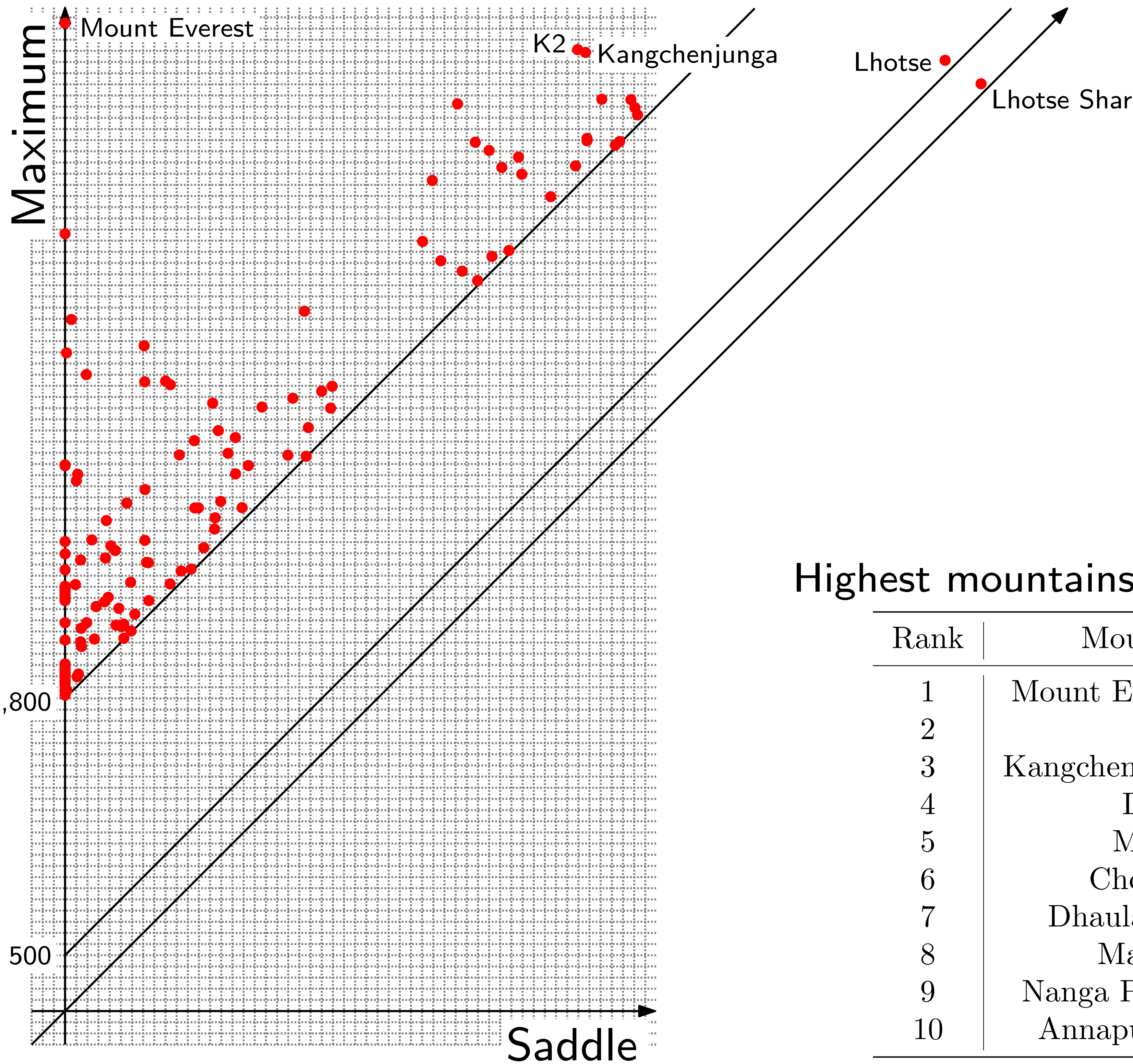


Persistence diagram records for each peak its value on the vertical axis, and the value of the saddle where it merges into a higher peak on the horizontal axis.

Persistence Diagram of Elevation on Earth



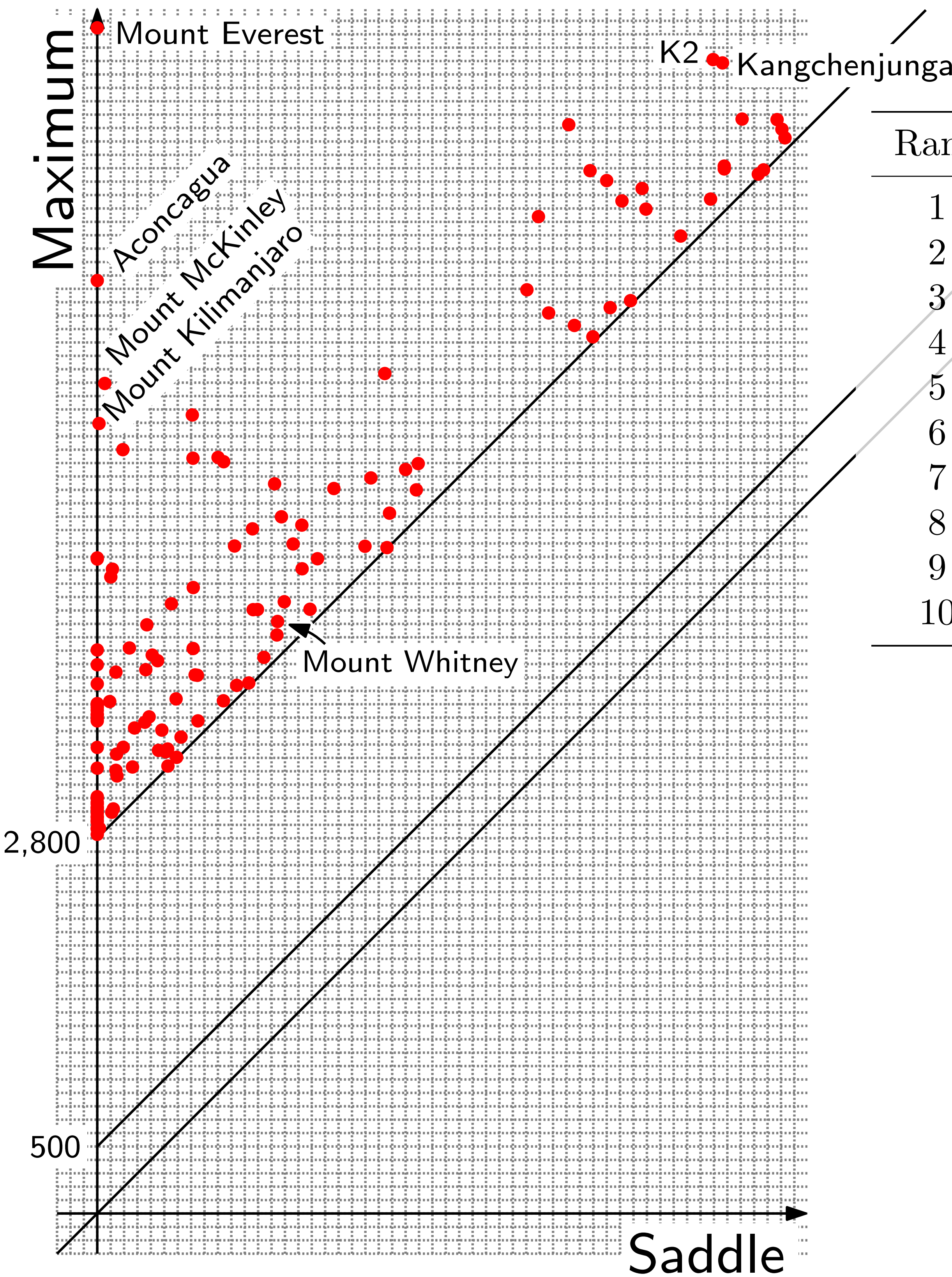
Persistence Diagram of Elevation on Earth



Highest mountains with prominence > 500 m.

Rank	Mountain	Height	Prominence
1	Mount Everest	8,848	8,848
2	K2	8,611	4,017
3	Kangchenjunga	8,586	3,922
4	Lhotse	8,516	610
5	Makalu	8,485	2,386
6	Cho Oyu	8,188	2,340
7	Dhaulagiri I	8,167	3,357
8	Manaslu	8,163	3,092
9	Nanga Parbat	8,126	4,608
10	Annapurna I	8,091	2,984

Persistence Diagram of Elevation on Earth



Mountains with highest prominence.

Rank	Mountain	Height	Highest point of	Prominence
1	Mount Everest	8,848	World	8,848
2	Aconcagua	6,962	Americas	6,962
3	Mount McKinley	6,194	North America	6,138
4	Mount Kilimanjaro	5,895	Africa	5,882
5	Pico Cristbal Coln	5,700	in Colombia	5,509
6	Mount Logan	5,959	Canada	5,250
7	Pico de Orizaba	5,636	Mexico	4,922
8	Vinson Massif	4,892	Antarctica	4,892
9	Puncak Jaya	4,884	New Guinea	4,884
10	Mount Elbrus	5,642	Europe	4,741

Highest mountains with prominence $>$ 500 m.

Rank	Mountain	Height	Prominence
1	Mount Everest	8,848	8,848
2	K2	8,611	4,017
3	Kangchenjunga	8,586	3,922
4	Lhotse	8,516	610
5	Makalu	8,485	2,386
6	Cho Oyu	8,188	2,340
7	Dhaulagiri I	8,167	3,357
8	Manaslu	8,163	3,092
9	Nanga Parbat	8,126	4,608
10	Annapurna I	8,091	2,984

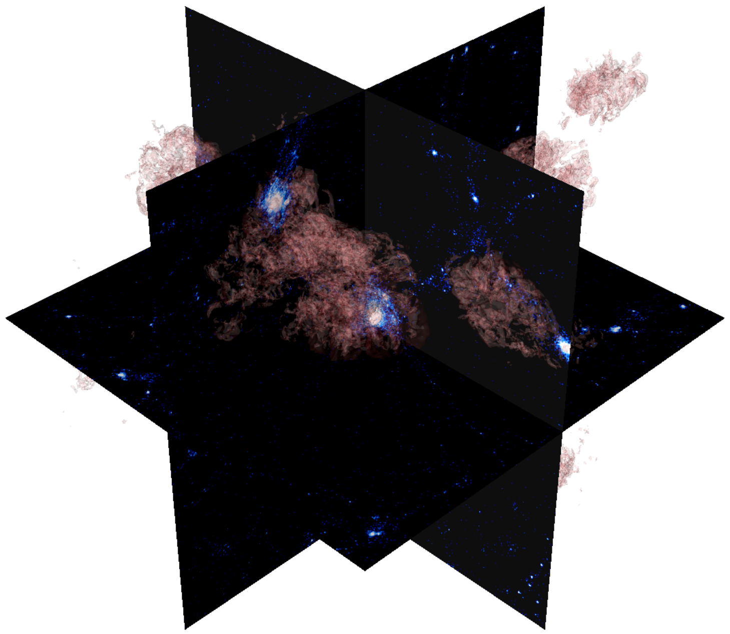
Motivation

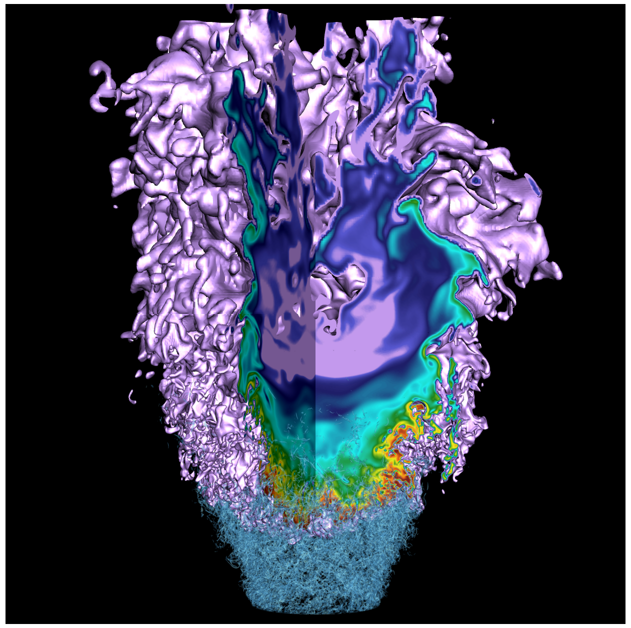
Natural phenomena modeled as scalar functions, $f : \mathbb{X} \to \mathbb{R}$

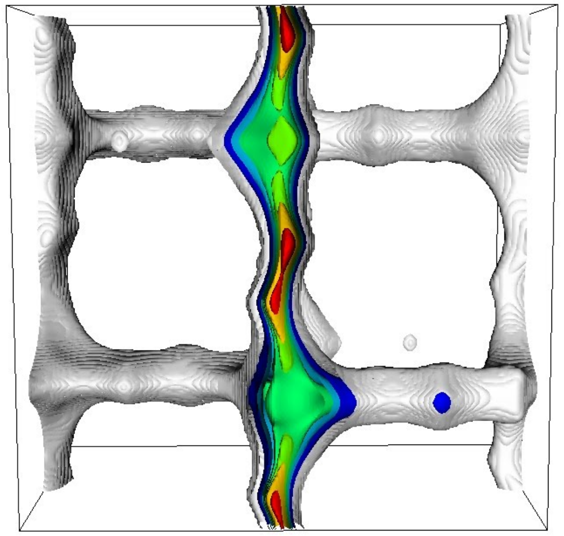
- density of galaxies
- rate of fuel consumption during combustion encodes a flame
- geometry of a material encoded in its distance function

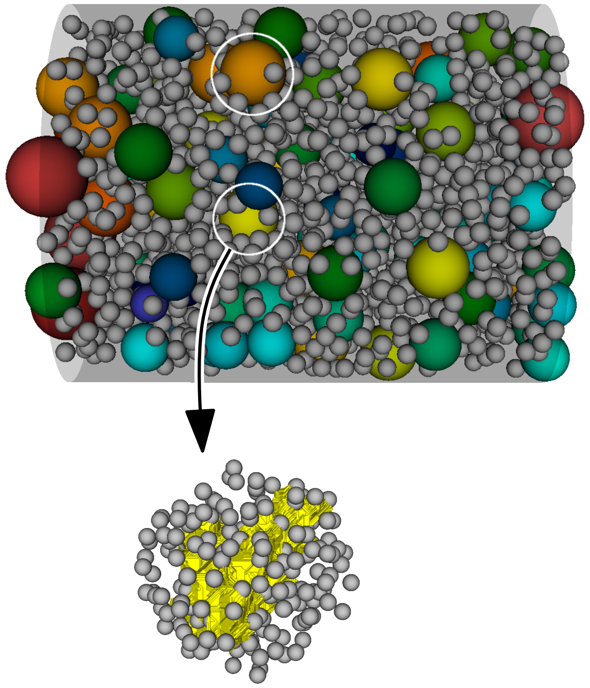
To analyze such data, need to detect and extract salient features; compute global statistics.

Topological features in scientific data:









(Source: CCSE, CCC, SCG at LBNL.)

Functions

Persistence is defined with respect to any scalar function $f : \mathbb{X} \to \mathbb{R}$.

if f is . . .	**persistent maxima capture significant . . .**
elevation on Earth	mountains

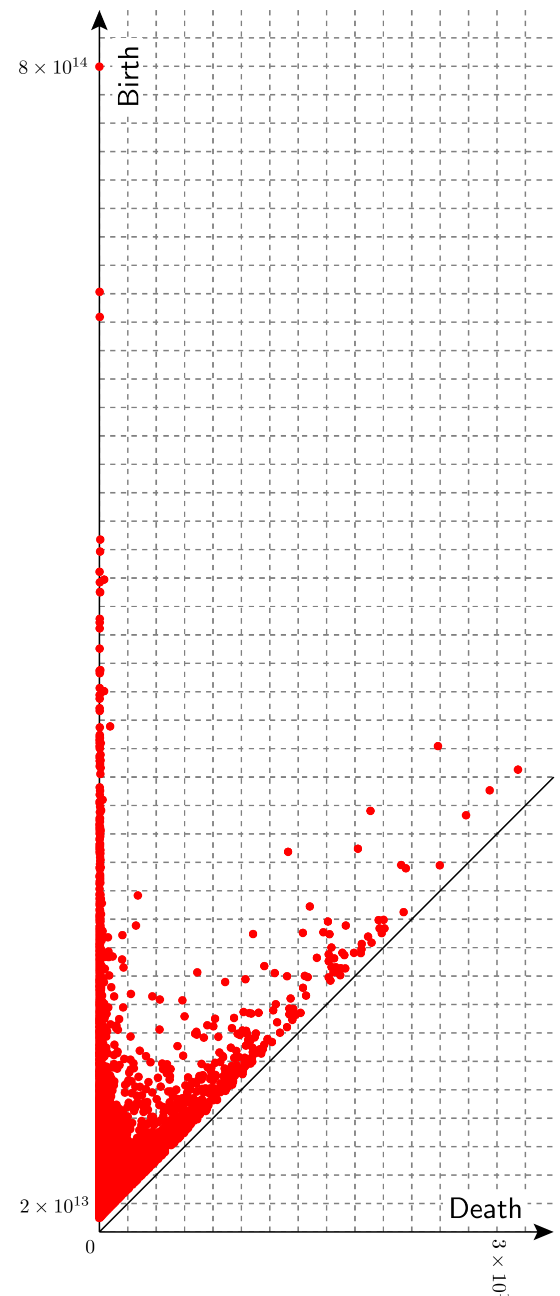
Functions

Persistence is defined with respect to any scalar function $f : \mathbb{X} \to \mathbb{R}$.

if f is . . . **persistent maxima capture significant . . .**

elevation on Earth — mountains

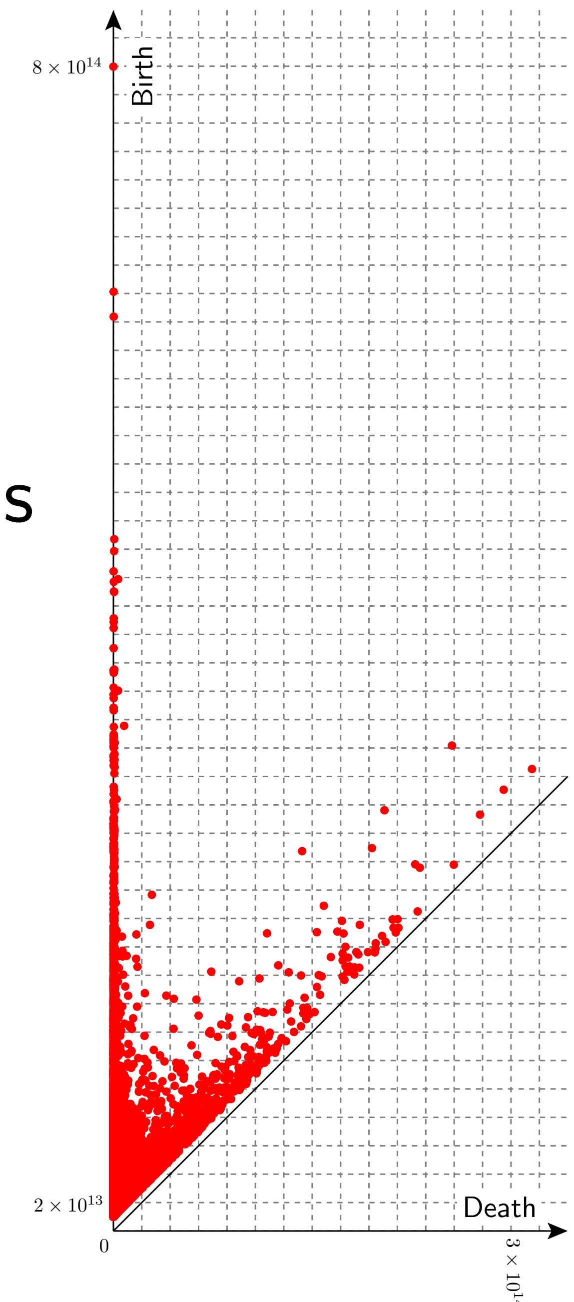
density of particles — clusters
e.g., halos in astrophysical data



Functions

Persistence is defined with respect to any scalar function $f : \mathbb{X} \to \mathbb{R}$.

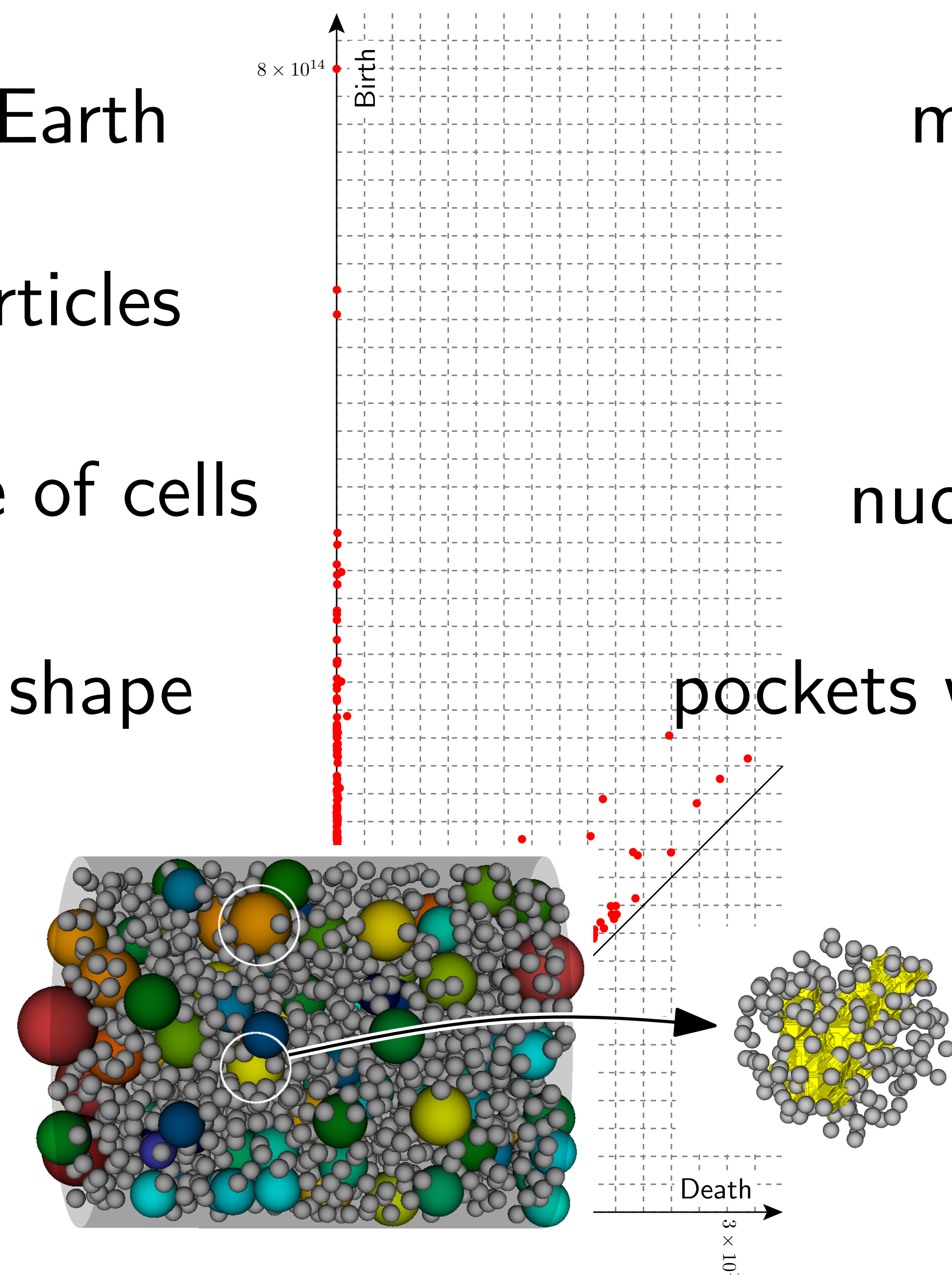
if f is ...	**persistent maxima capture significant ...**
elevation on Earth	mountains
density of particles	clusters e.g., halos in astrophysical data
grayscale image of cells	nucleii of cells



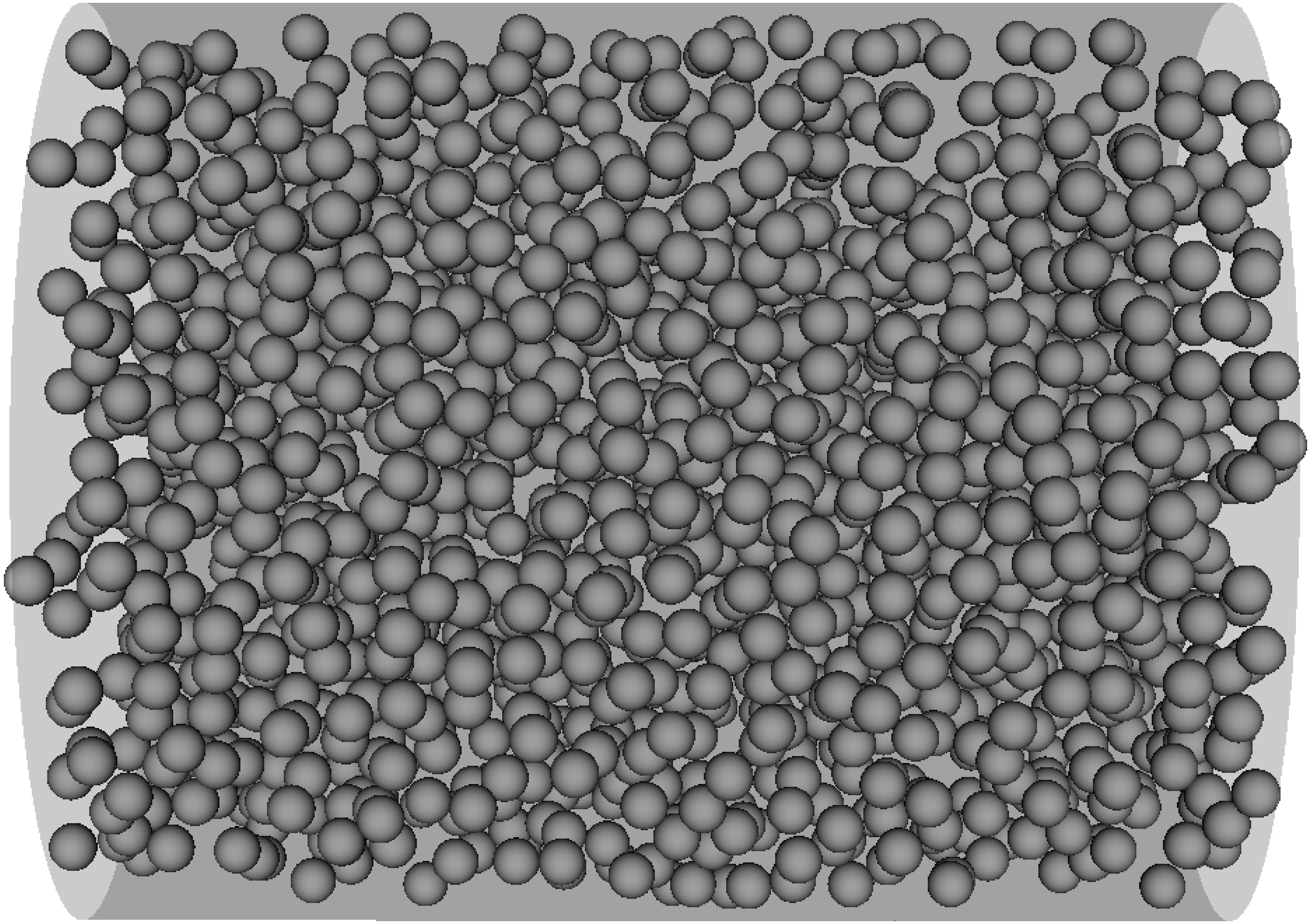
Functions

Persistence is defined with respect to any scalar function $f : \mathbb{X} \to \mathbb{R}$.

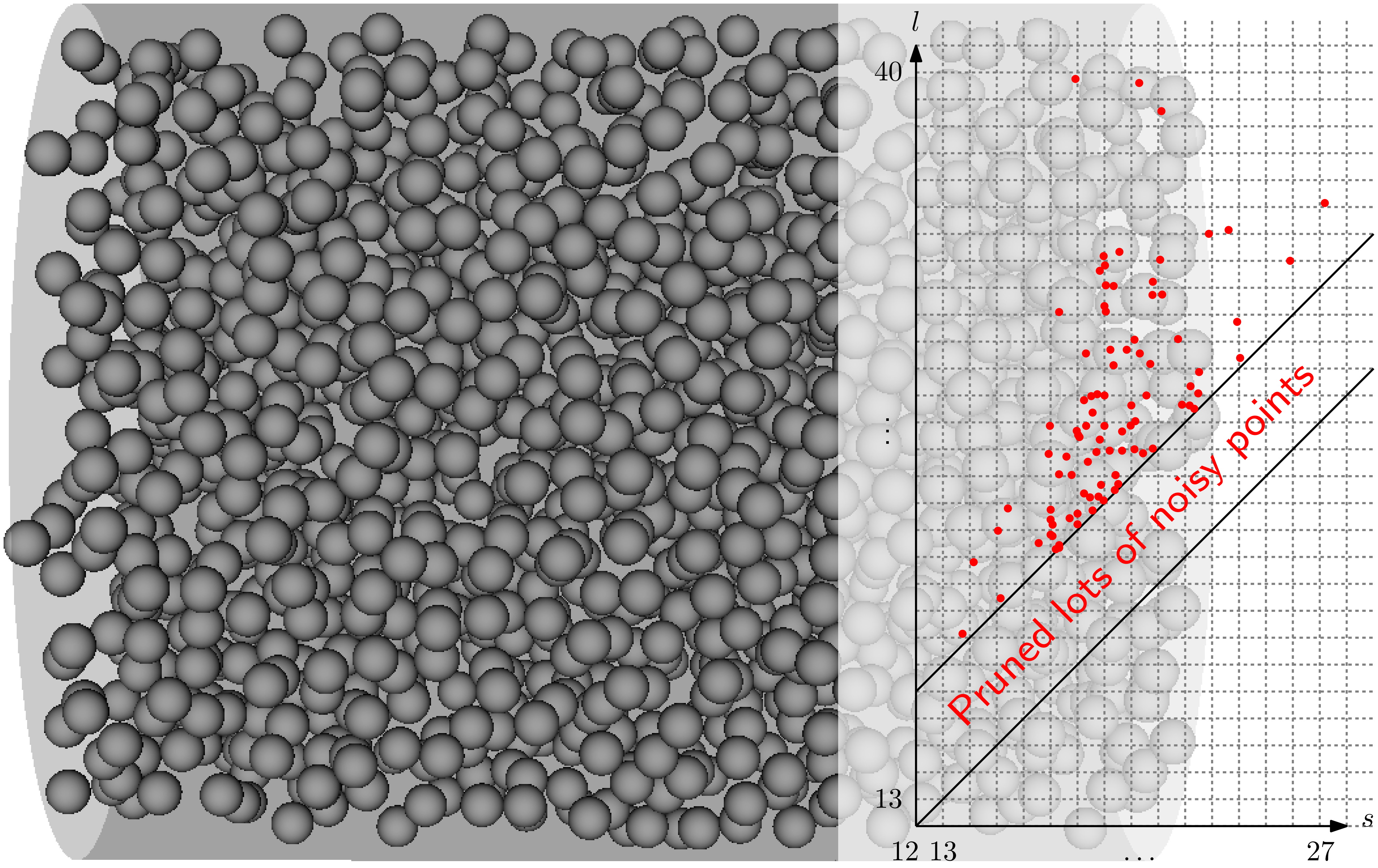
if f is ...	**persistent maxima capture significant ...**
elevation on Earth	mountains
density of particles	clusters e.g., halos in astrophysical data
grayscale image of cells	nucleii of cells
distance to a shape	pockets within the shape e.g., voids in a subsurface rock formation, or in a protein



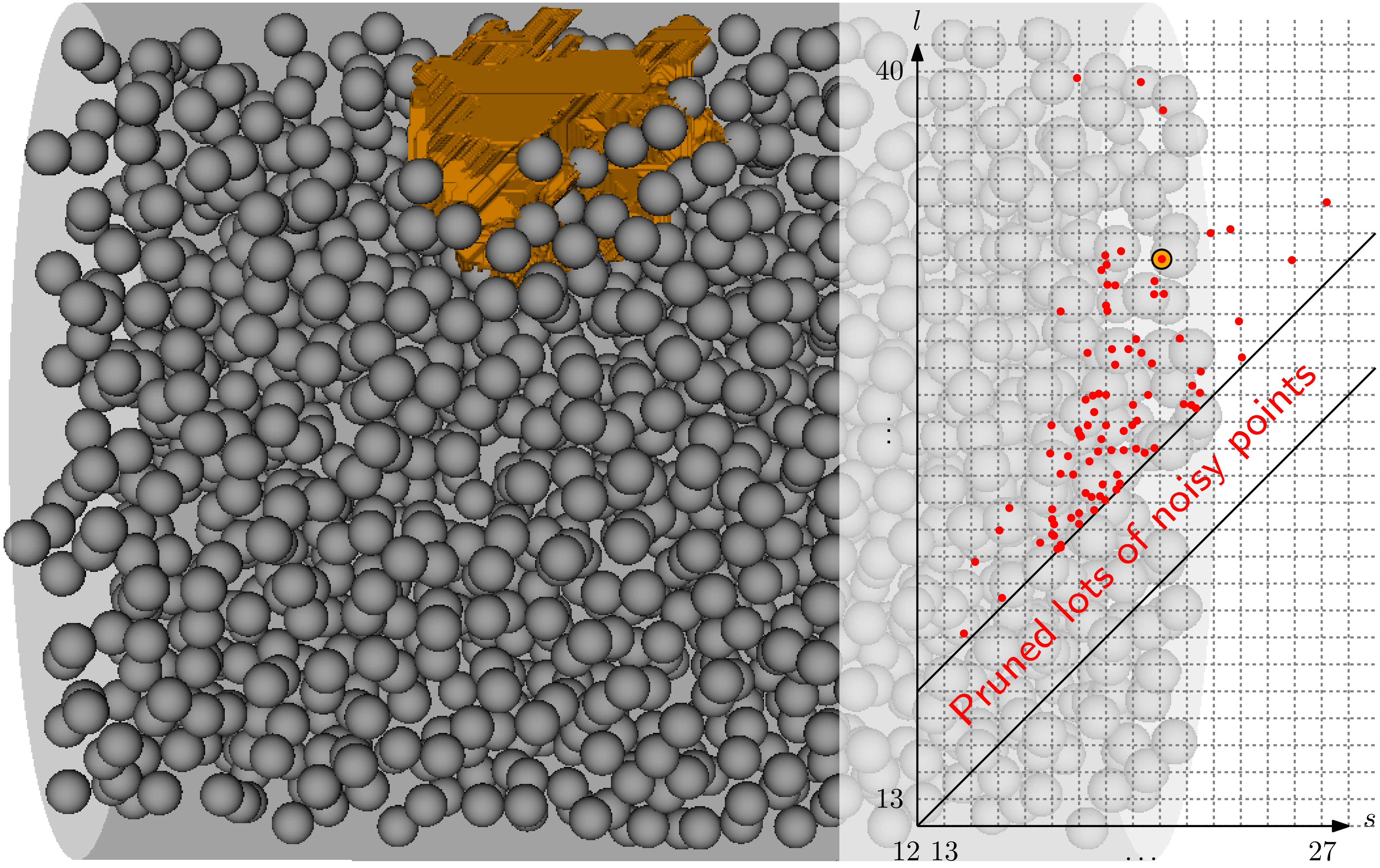
Pockets



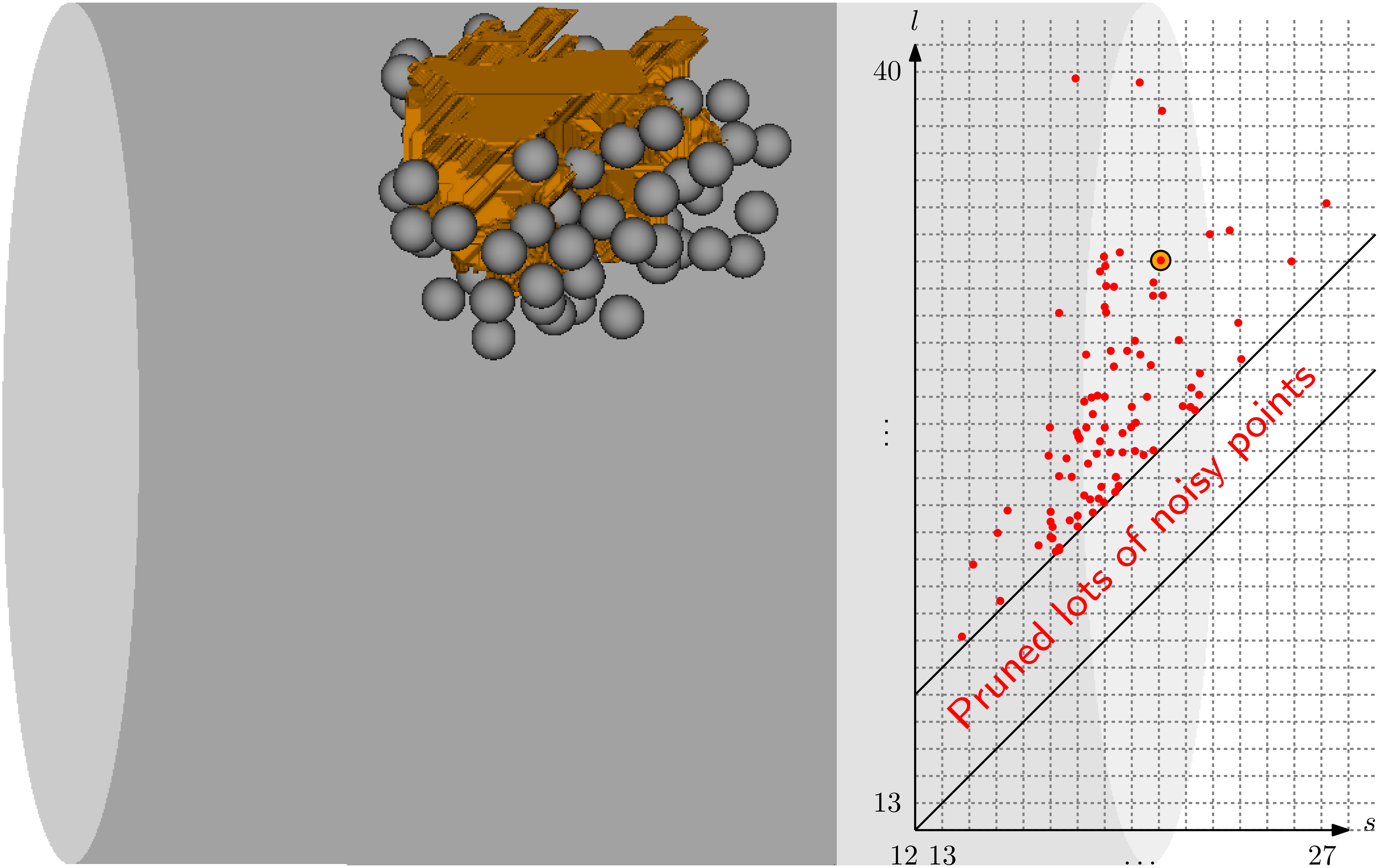
Pockets



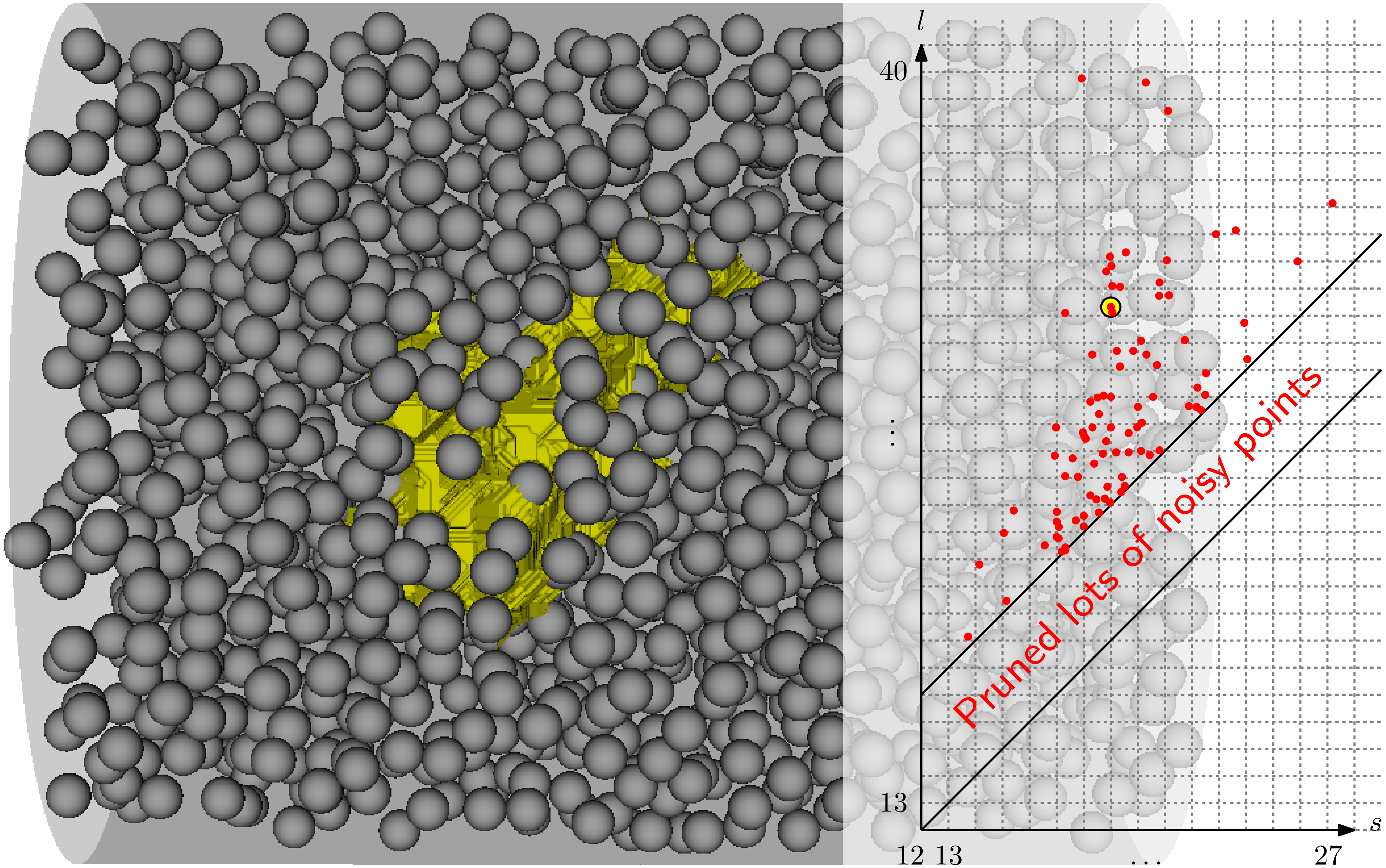
Pockets



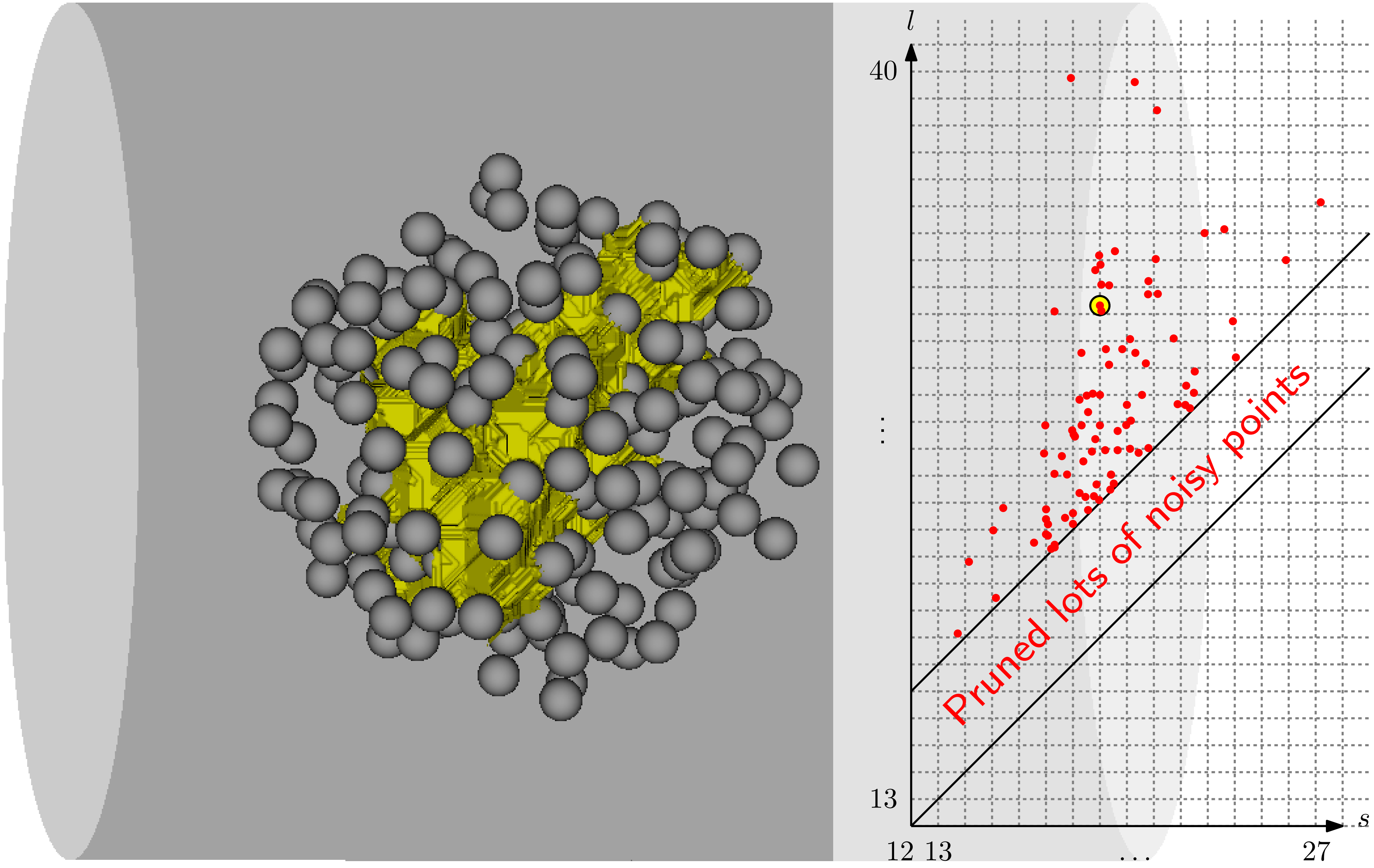
Pockets



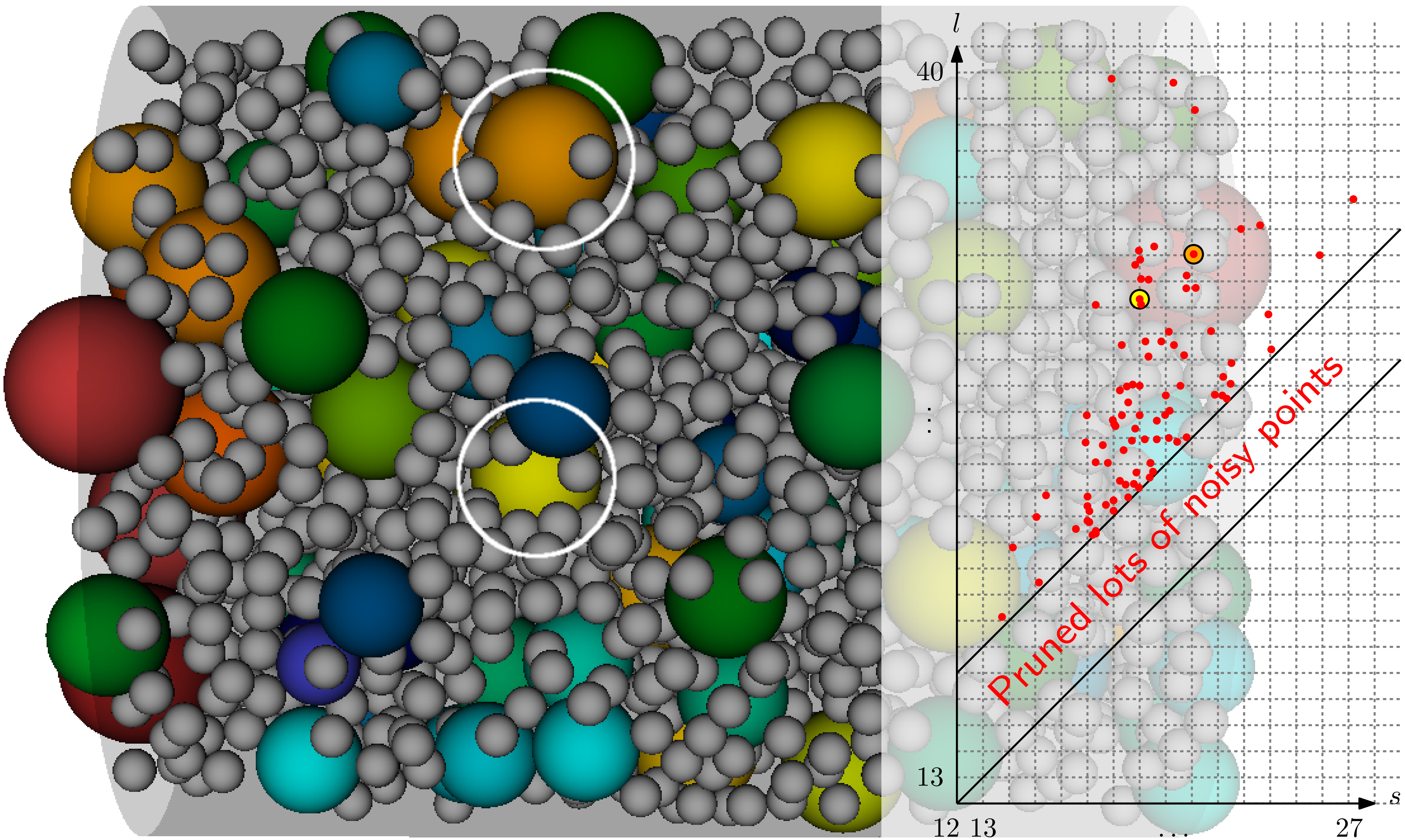
Pockets



Pockets



Pockets

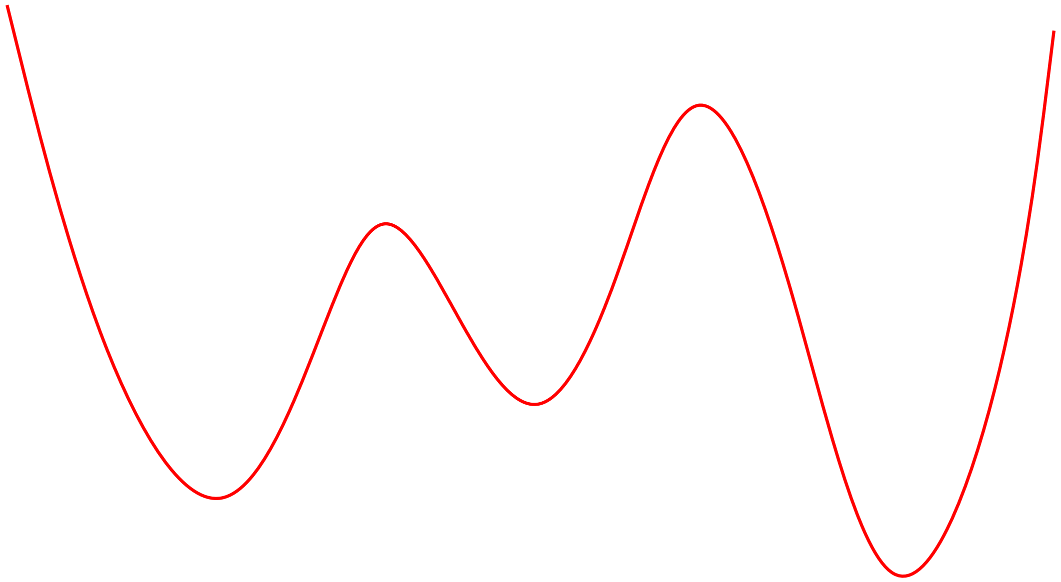


Merge Trees

Function: $f : \mathbb{X} \to \mathbb{R}$

Sublevel set: $\mathbb{X}_a = f^{-1}(-\infty, a]$

Merge tree = record connectivity of the components of sublevel sets

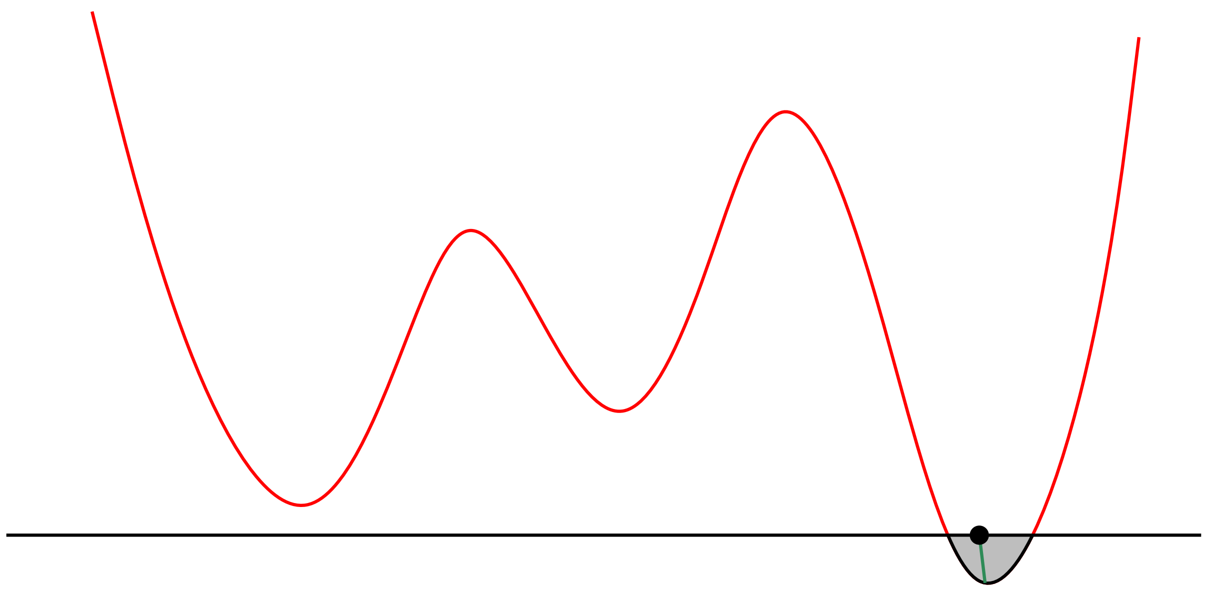


Merge Trees

Function: $f : \mathbb{X} \to \mathbb{R}$

Sublevel set: $\mathbb{X}_a = f^{-1}(-\infty, a]$

Merge tree = record connectivity of the components of sublevel sets

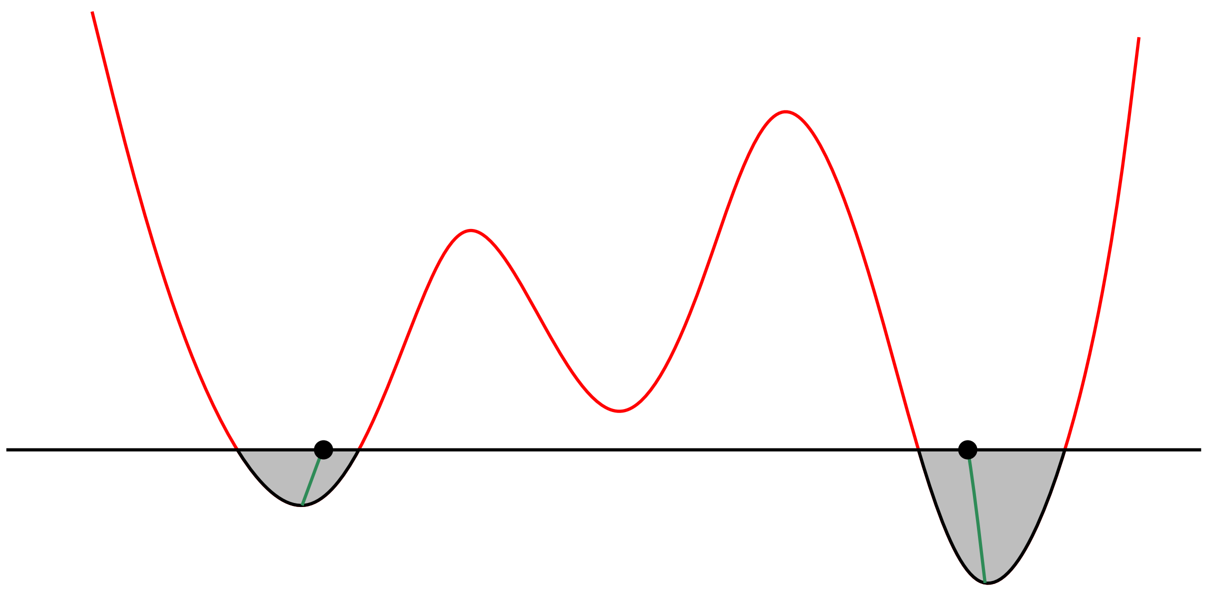


Merge Trees

Function: $f : \mathbb{X} \to \mathbb{R}$

Sublevel set: $\mathbb{X}_a = f^{-1}(-\infty, a]$

Merge tree = record connectivity of the components of sublevel sets

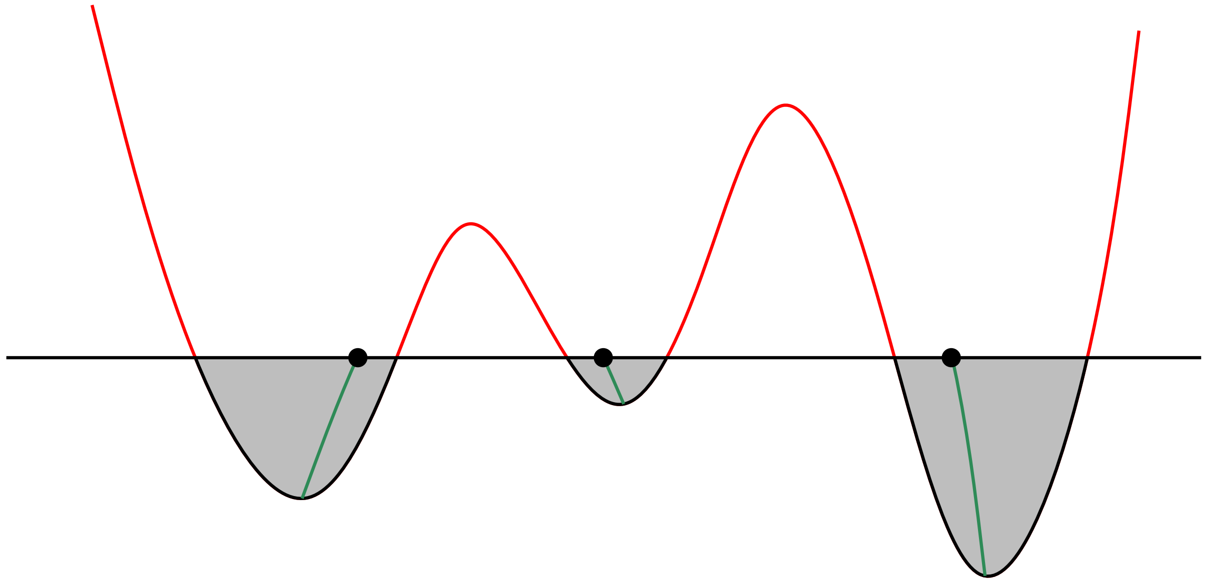


Merge Trees

Function: $f : \mathbb{X} \to \mathbb{R}$

Sublevel set: $\mathbb{X}_a = f^{-1}(-\infty, a]$

Merge tree = record connectivity of the components of sublevel sets

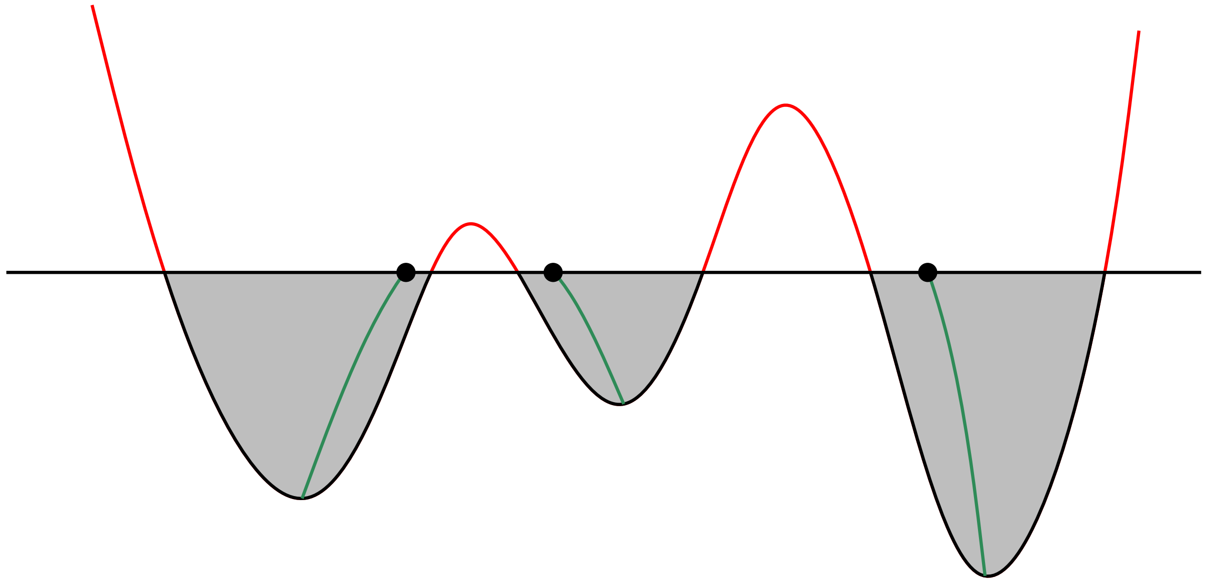


Merge Trees

Function: $f : \mathbb{X} \to \mathbb{R}$

Sublevel set: $\mathbb{X}_a = f^{-1}(-\infty, a]$

Merge tree = record connectivity of the components of sublevel sets

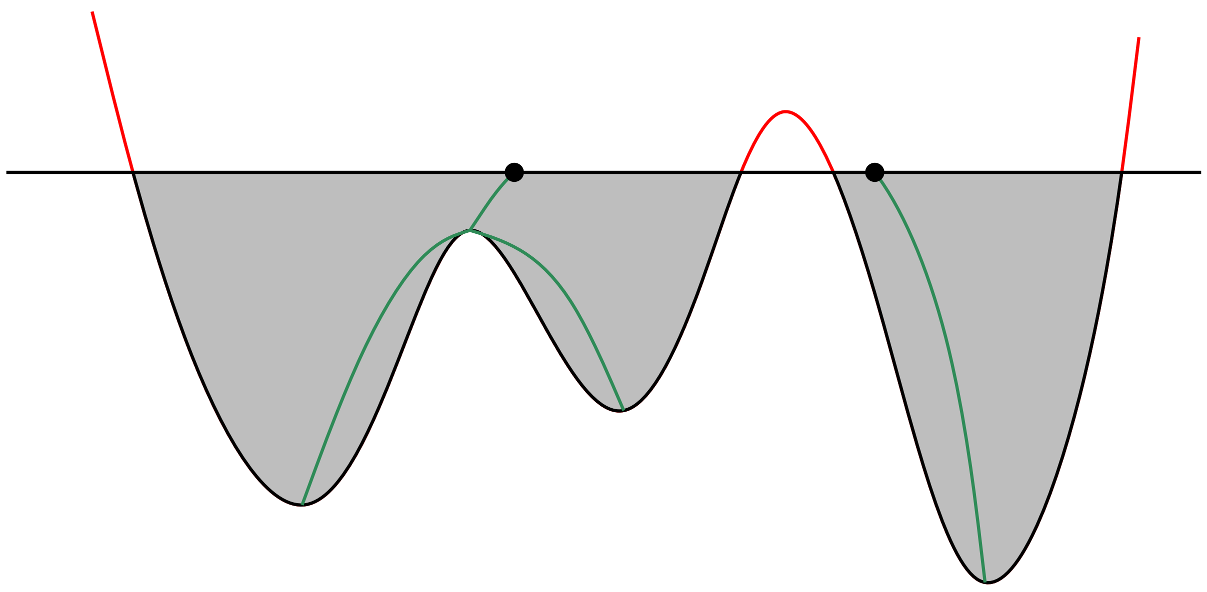


Merge Trees

Function: $f : \mathbb{X} \to \mathbb{R}$

Sublevel set: $\mathbb{X}_a = f^{-1}(-\infty, a]$

Merge tree = record connectivity of the components of sublevel sets

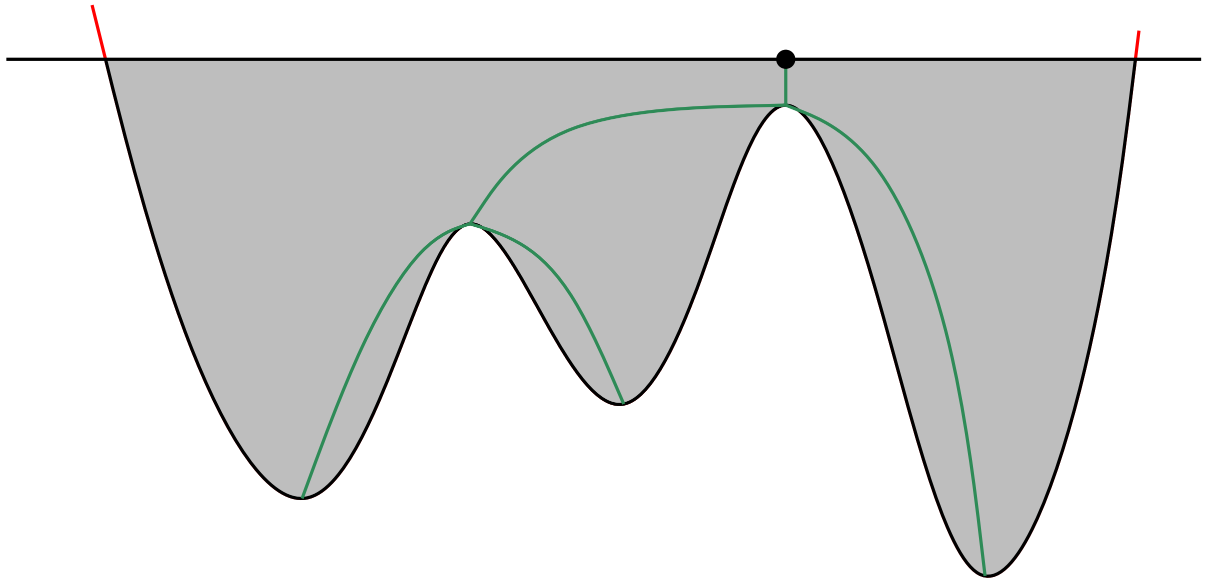


Merge Trees

Function: $f : \mathbb{X} \to \mathbb{R}$

Sublevel set: $\mathbb{X}_a = f^{-1}(-\infty, a]$

Merge tree = record connectivity of the components of sublevel sets

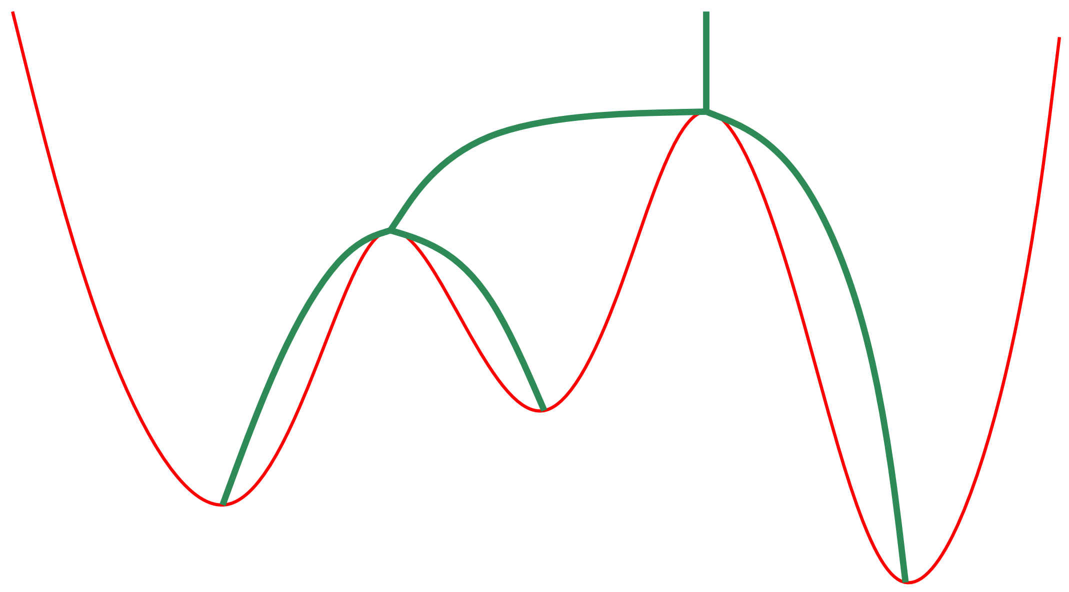


Merge Trees

Function: $f : \mathbb{X} \to \mathbb{R}$

Sublevel set: $\mathbb{X}_a = f^{-1}(-\infty, a]$

Merge tree = record connectivity of the components of sublevel sets

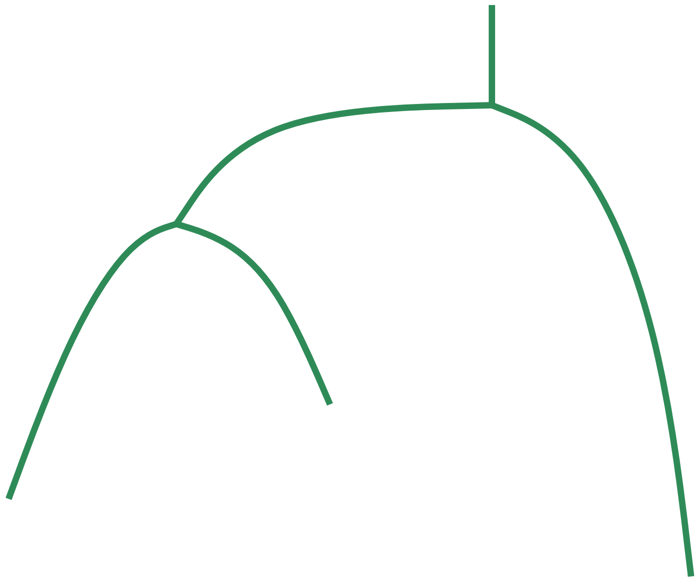


Merge Trees

Function: $f : \mathbb{X} \to \mathbb{R}$

Sublevel set: $\mathbb{X}_a = f^{-1}(-\infty, a]$

Merge tree = record connectivity of the components of sublevel sets

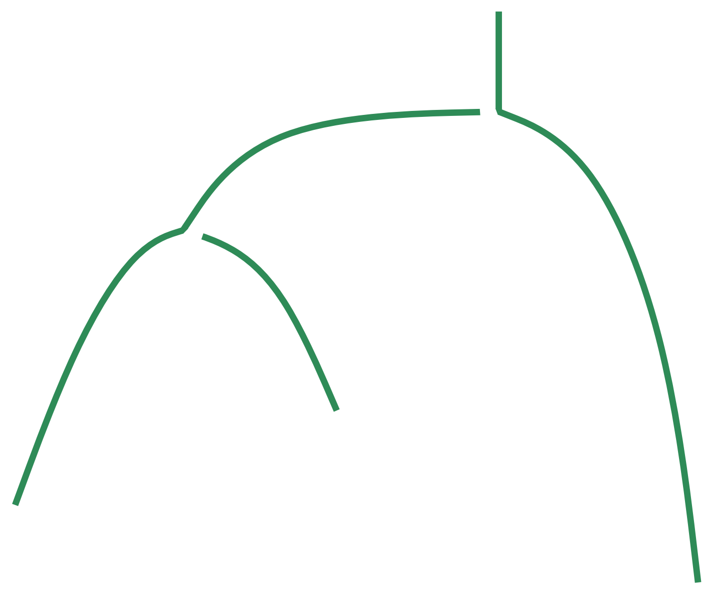


Merge Trees

Function: $f : \mathbb{X} \to \mathbb{R}$

Sublevel set: $\mathbb{X}_a = f^{-1}(-\infty, a]$

Merge tree = record connectivity of the components of sublevel sets

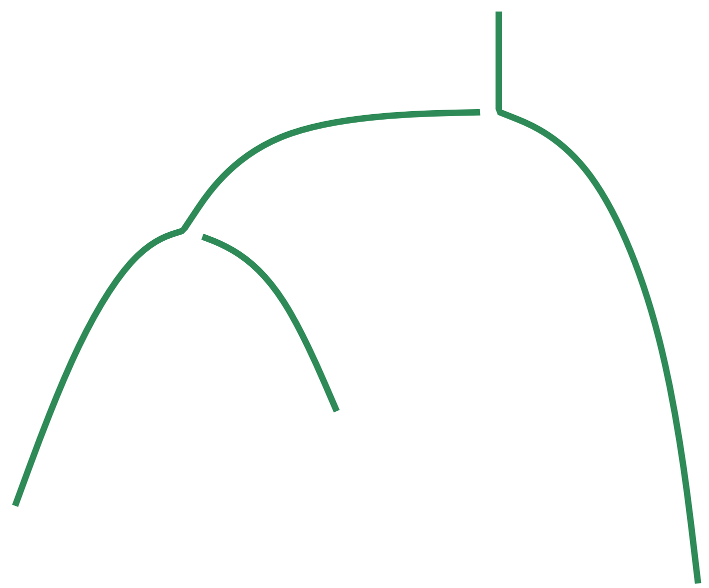


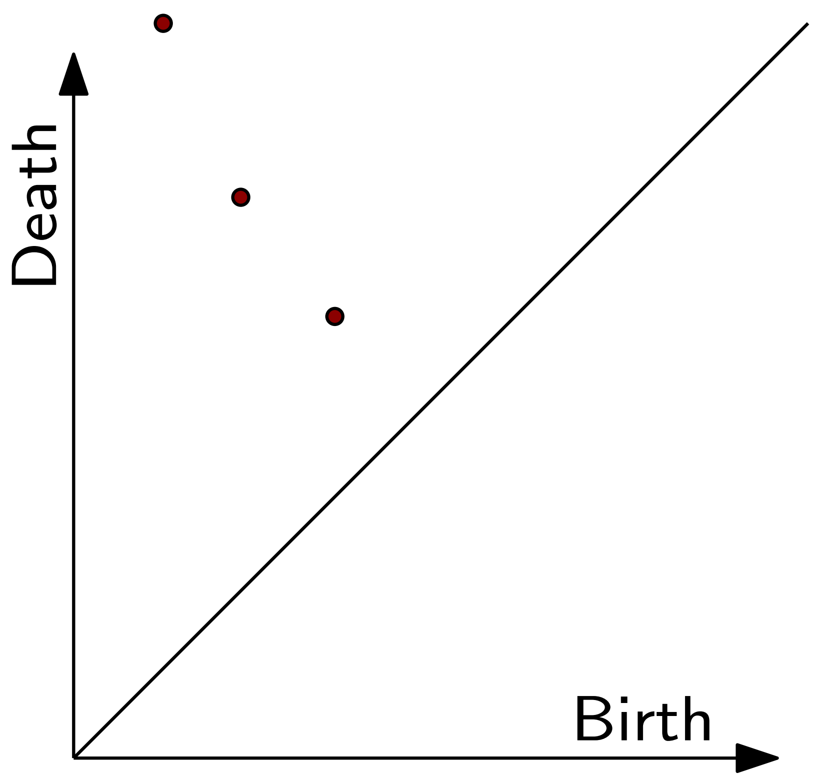
Merge Trees

Function: $f : \mathbb{X} \to \mathbb{R}$

Sublevel set: $\mathbb{X}_a = f^{-1}(-\infty, a]$

Merge tree = record connectivity of the components of sublevel sets



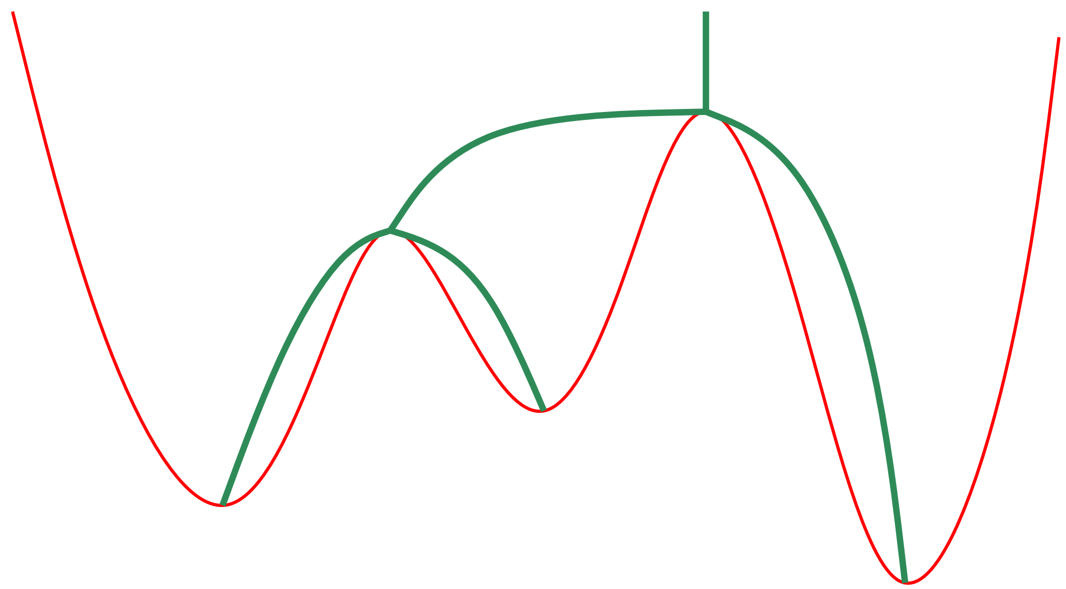


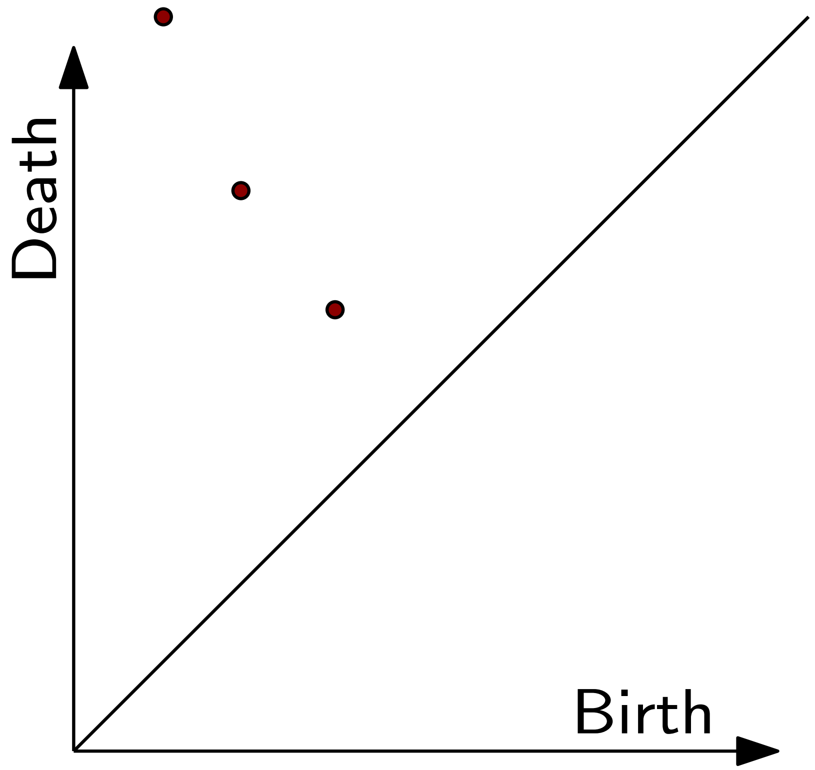
Merge Trees

Function: $f : \mathbb{X} \to \mathbb{R}$

Sublevel set: $\mathbb{X}_a = f^{-1}(-\infty, a]$

Merge tree = record connectivity of the components of sublevel sets



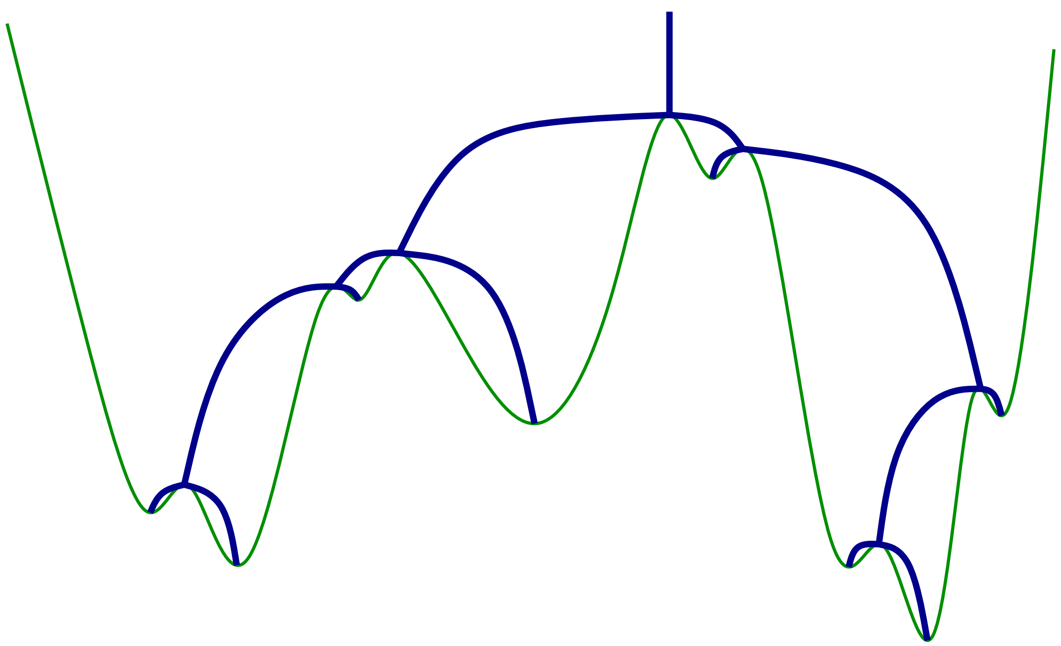


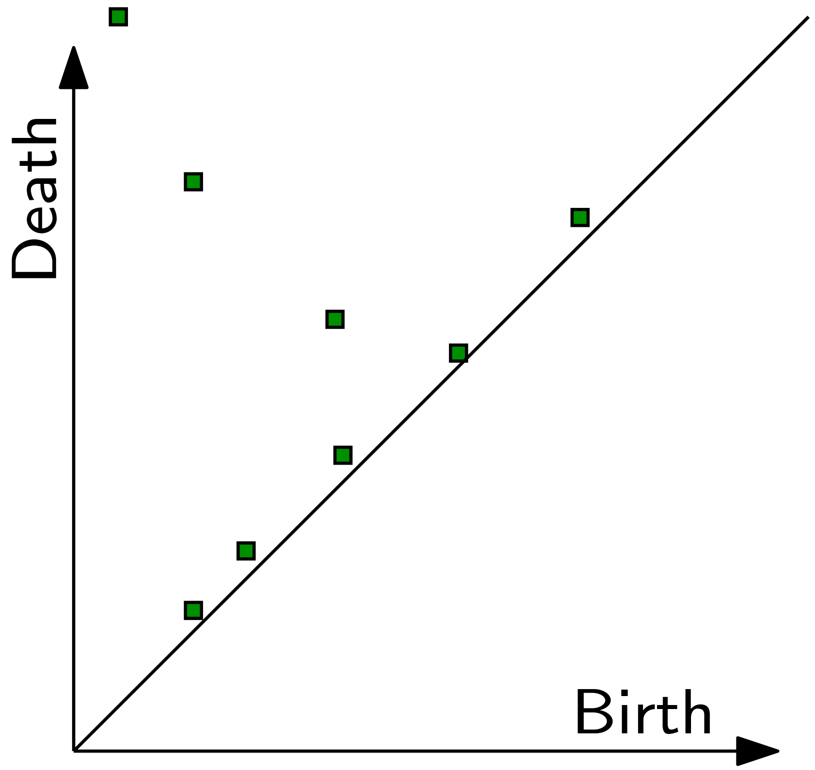
Merge Trees

Function: $f : \mathbb{X} \to \mathbb{R}$

Sublevel set: $\mathbb{X}_a = f^{-1}(-\infty, a]$

Merge tree = record connectivity of the components of sublevel sets



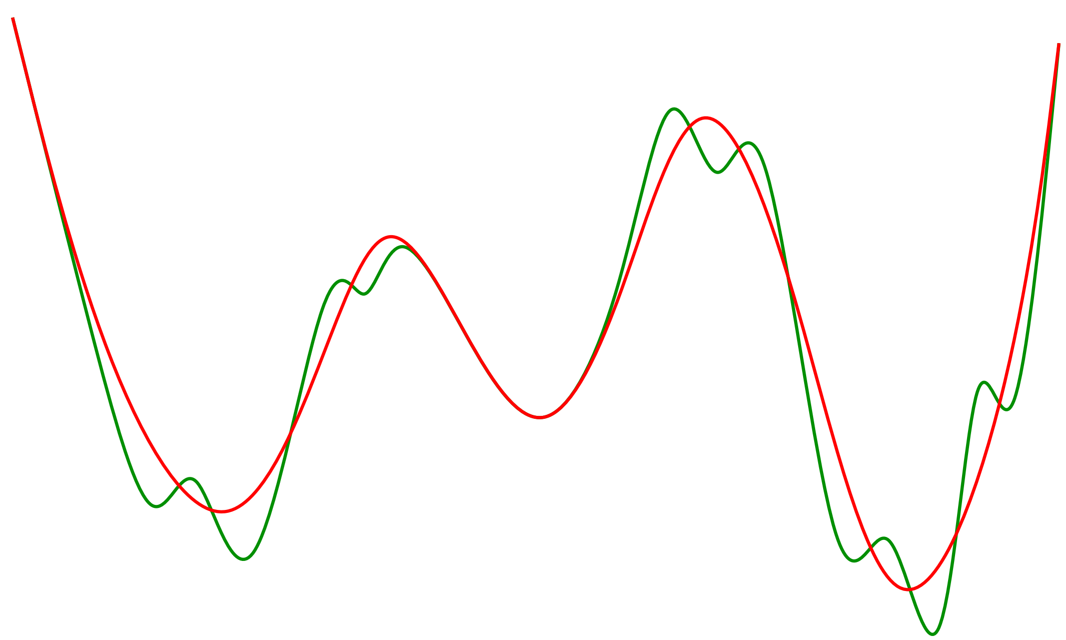


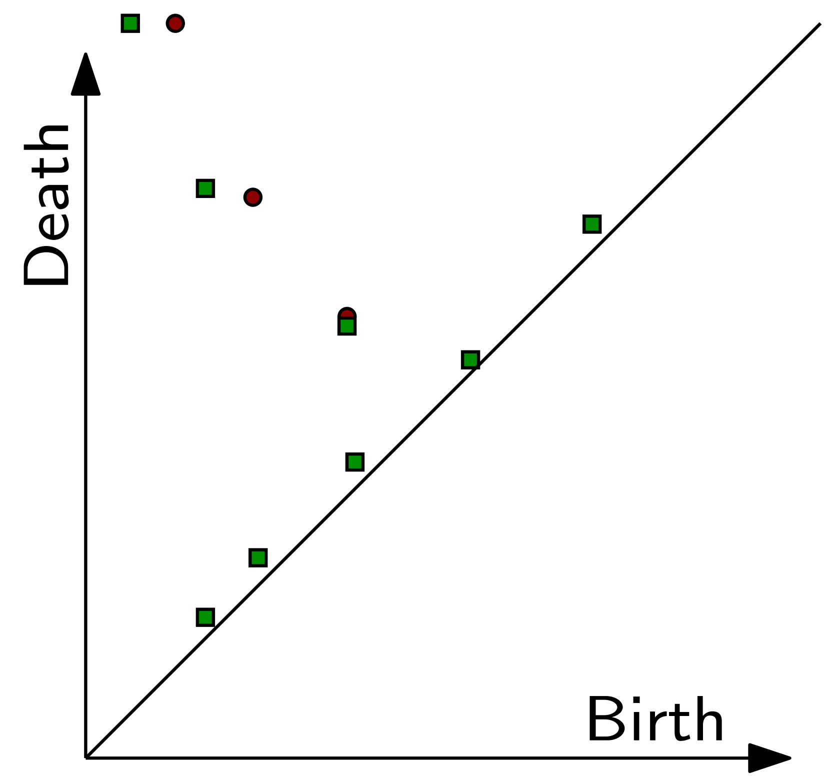
Merge Trees

Function: $f : \mathbb{X} \to \mathbb{R}$

Sublevel set: $\mathbb{X}_a = f^{-1}(-\infty, a]$

Merge tree $=$ record connectivity of the components of sublevel sets





Stability Theorem (for persistence diagrams):

$$\mathrm{d_B}(\mathrm{Dgm}(f), \mathrm{Dgm}(g)) \leq \|f - g\|_\infty.$$

Interleaving Distance

between Merge Trees

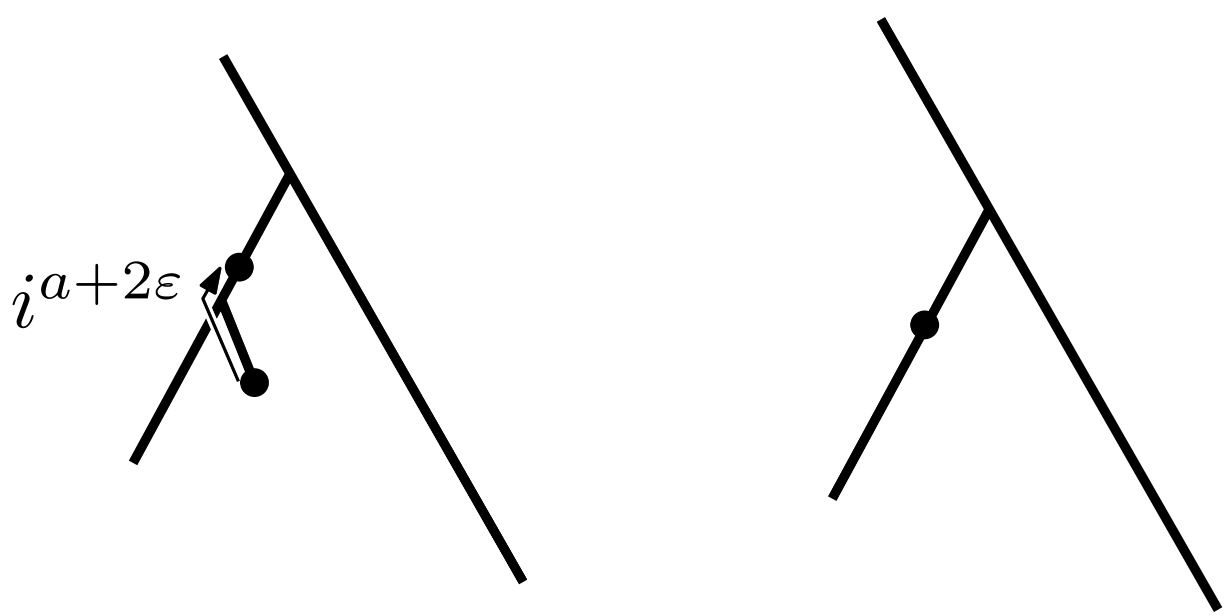
Interleaving Distance

Trees T_f and T_g.

$\hat{f} : \mathrm{T}_f \to \mathbb{R}$
$\hat{g} : \mathrm{T}_g \to \mathbb{R}$

$i^{2\varepsilon}$ shift map in T_f
$j^{2\varepsilon}$ shift map in T_g

(Inclusion of component F_x into a component of $F_{x+2\varepsilon}$.)



Interleaving Distance

Trees T_f and T_g.

$$\alpha^\varepsilon : \mathrm{T}_f \to \mathrm{T}_g \qquad \hat{f} : \mathrm{T}_f \to \mathbb{R}$$
$$\beta^\varepsilon : \mathrm{T}_g \to \mathrm{T}_f \qquad \hat{g} : \mathrm{T}_g \to \mathbb{R}$$

$i^{2\varepsilon}$ shift map in T_f

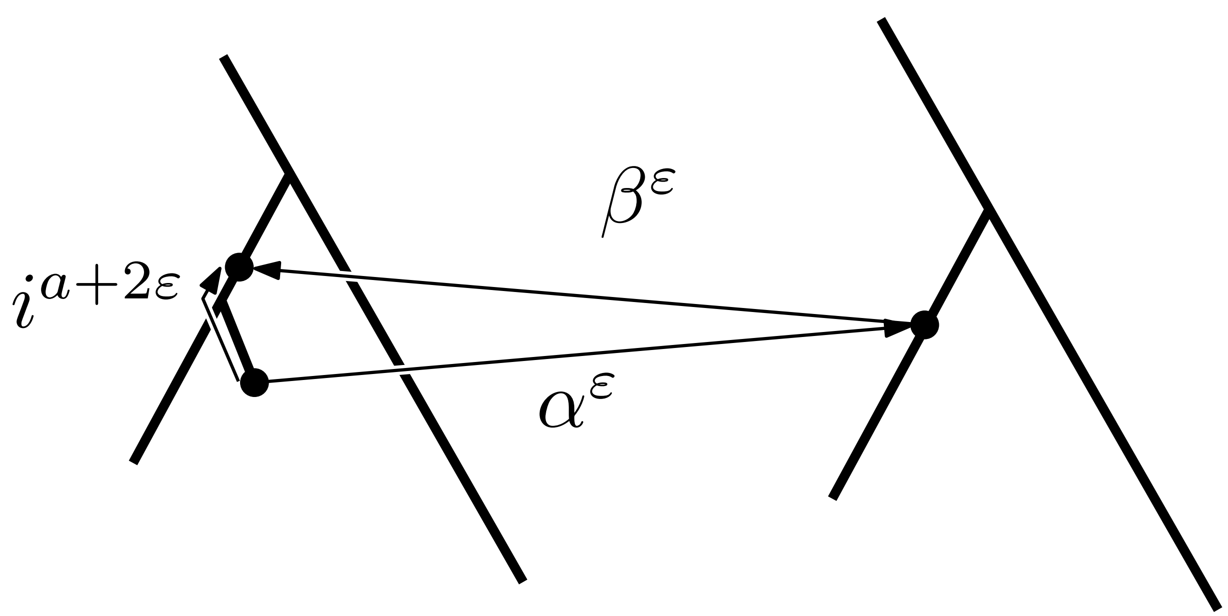
$j^{2\varepsilon}$ shift map in T_g

(Inclusion of component F_x into a component of $F_{x+2\varepsilon}$.)

α^ε and β^ε are ε-**compatible**.

$$\hat{g}(\alpha^\varepsilon(x)) = \hat{f}(x) + \varepsilon \qquad \hat{f}(\beta^\varepsilon(x)) = \hat{g}(x) + \varepsilon$$

$$\beta^\varepsilon \circ \alpha^\varepsilon = i^{2\varepsilon} \qquad \alpha^\varepsilon \circ \beta^\varepsilon = j^{2\varepsilon}$$



Interleaving Distance

Trees T_f and T_g.

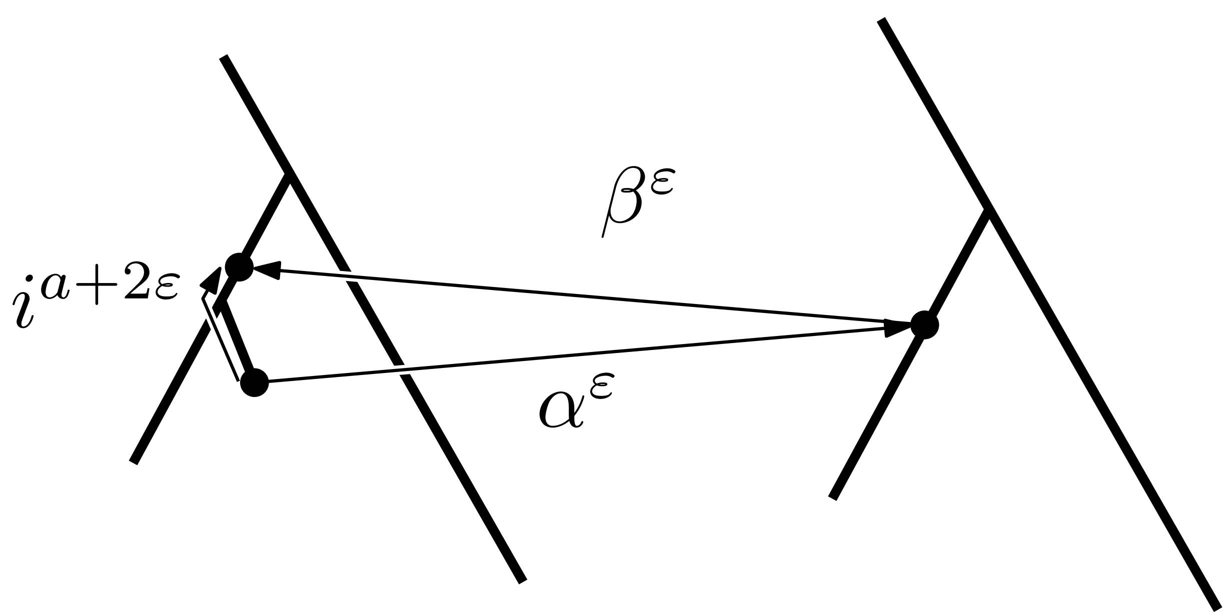
$$\mathrm{d_I}(\mathrm{T}_f, \mathrm{T}_g) = \inf\{\varepsilon \mid \text{there are } \varepsilon\text{-compatible maps } \alpha^\varepsilon \text{ and } \beta^\varepsilon\}$$

(Inclusion of component F_x into a component of $F_{x+2\varepsilon}$.)

α^ε and β^ε are ε-**compatible**.

$$\hat{g}(\alpha^\varepsilon(x)) = \hat{f}(x) + \varepsilon \qquad \hat{f}(\beta^\varepsilon(x)) = \hat{g}(x) + \varepsilon$$

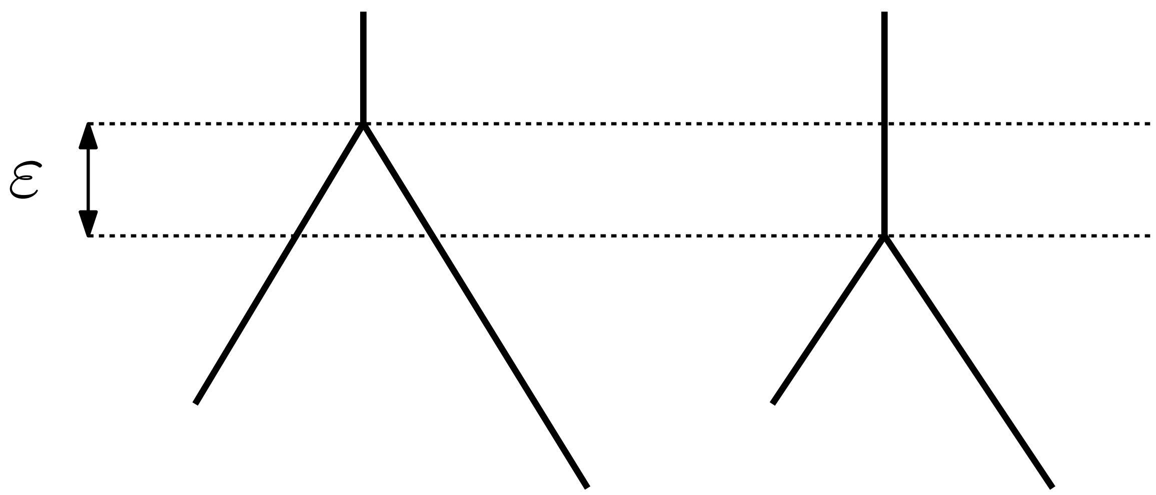
$$\beta^\varepsilon \circ \alpha^\varepsilon = i^{2\varepsilon} \qquad \alpha^\varepsilon \circ \beta^\varepsilon = j^{2\varepsilon}$$



Examples

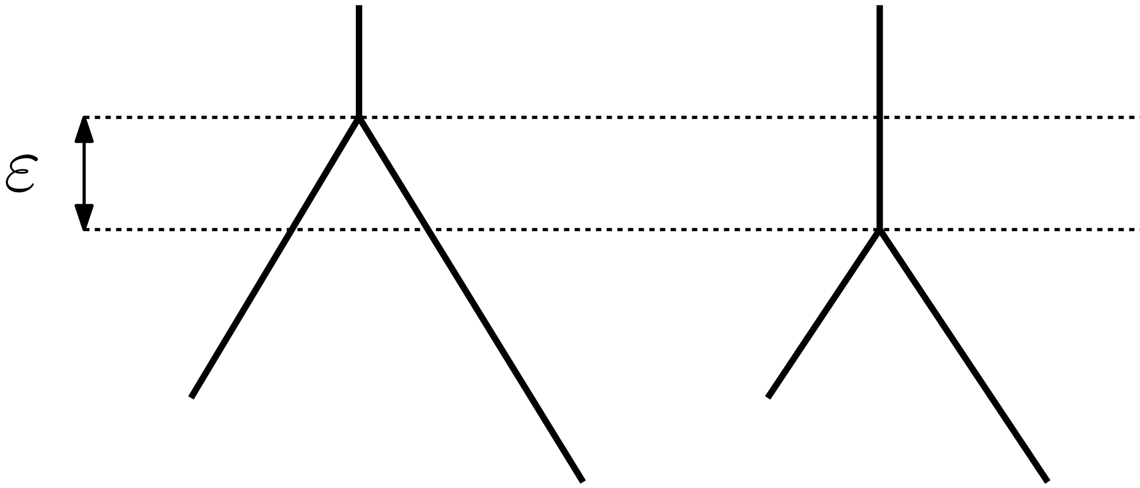
Examples

Shifted saddle: $\mathrm{d}_{\mathrm{I}} = \varepsilon$.

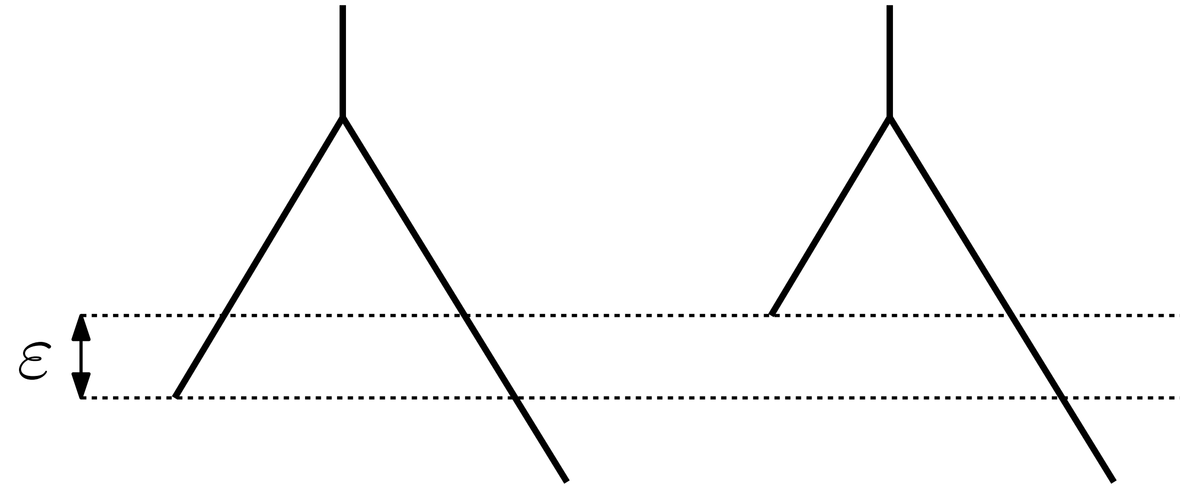


Examples

Shifted saddle: $\mathrm{d_I} = \varepsilon$.

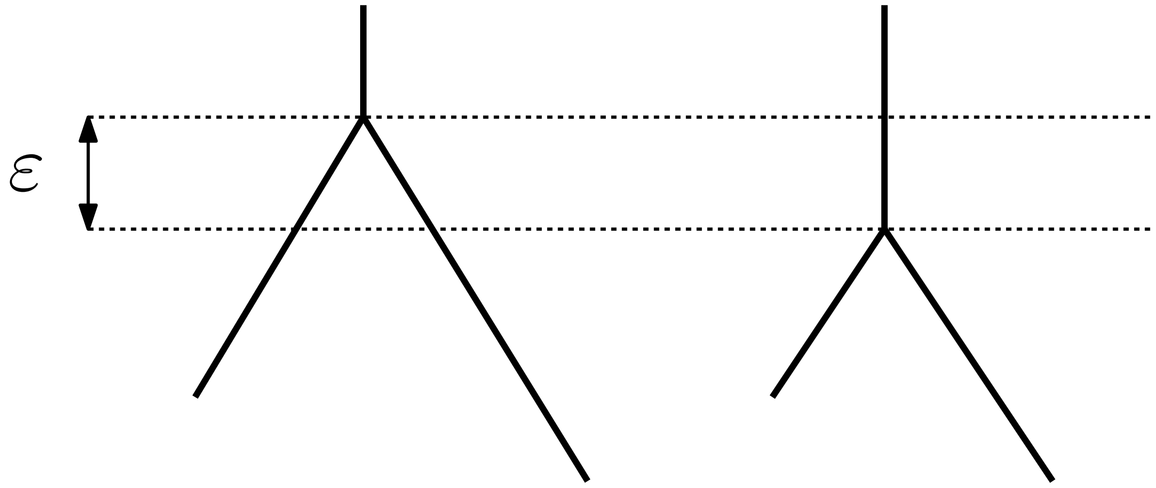


Shifted leaf: $\mathrm{d_I} = \varepsilon$.

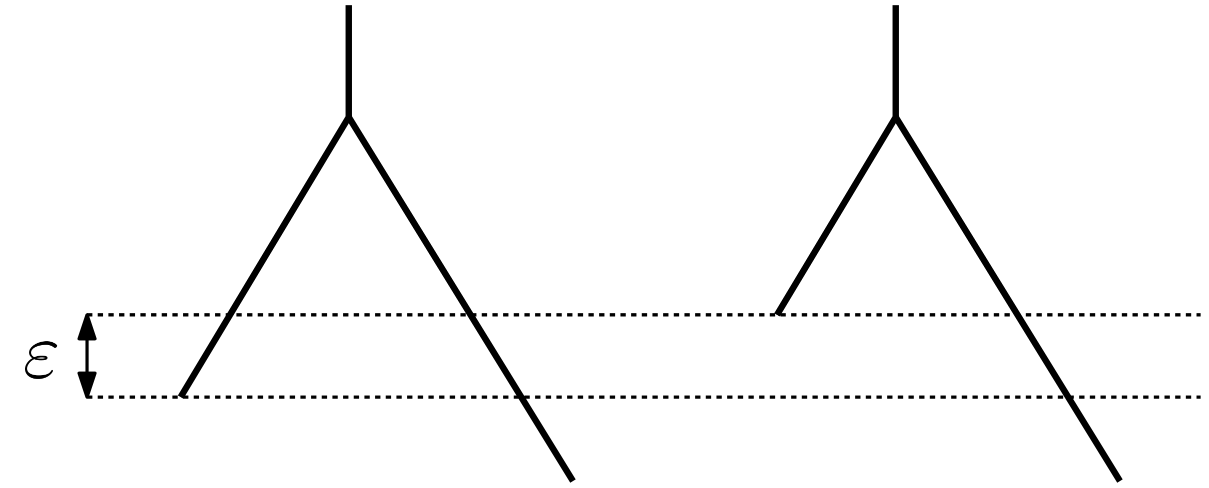


Examples

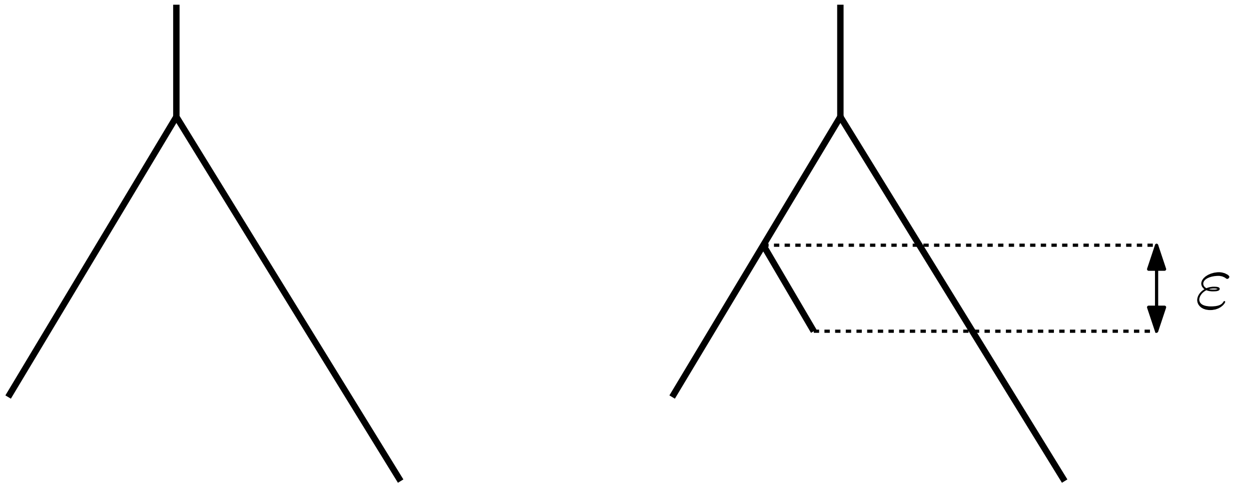
Shifted saddle: $\mathrm{d_I} = \varepsilon$.



Shifted leaf: $\mathrm{d_I} = \varepsilon$.



Missing branch: $\mathrm{d_I} = \varepsilon/2$.



$\mathrm{d_I}$ is a metric

1. $\mathrm{d_I}(\mathrm{T}, \mathrm{T}) = 0$;
2. $\mathrm{d_I}(\mathrm{T}_f, \mathrm{T}_g) = \mathrm{d_I}(\mathrm{T}_g, \mathrm{T}_f)$;
3. $\mathrm{d_I}(\mathrm{T}_1, \mathrm{T}_3) \leq \mathrm{d_I}(\mathrm{T}_1, T_2) + \mathrm{d_I}(T_2, T_3)$.

Proof:

1. $\alpha^0 = \beta^0 = \mathrm{Id}$;
2. symmetry of the definition;
3. $\alpha_{13} = \alpha_{12} \circ \alpha_{23}$; $\beta_{13} = \beta_{12} \circ \beta_{23}$.

Stability

Stability Theorem: $\mathrm{d_I}(\mathrm{T}_f, \mathrm{T}_g) \leq \|f - g\|_\infty$.

Stability

Stability Theorem: $\mathrm{d_I}(\mathrm{T}_f, \mathrm{T}_g) \leq \|f - g\|_\infty$.

Let $\varepsilon = \|f - g\|_\infty$.

$F_a \subseteq G_{a+\varepsilon} \subseteq F_{a+2\varepsilon}$.

$$F_a = f^{-1}(-\infty, a]$$

$$G_a = g^{-1}(-\infty, a]$$

Stability

Stability Theorem: $\mathrm{d_I}(\mathrm{T}_f, \mathrm{T}_g) \leq \|f - g\|_\infty$.

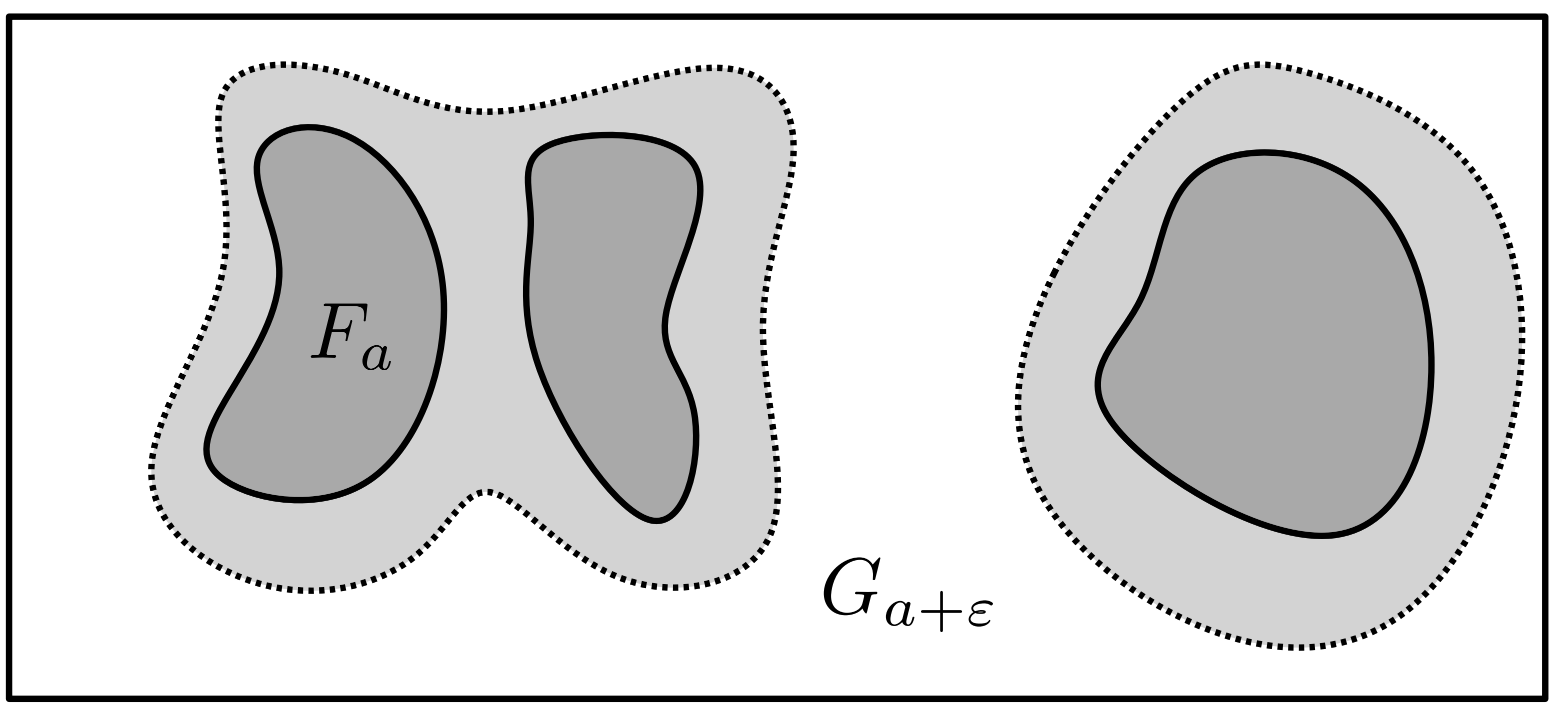
Let $\varepsilon = \|f - g\|_\infty$.

$F_a \subseteq G_{a+\varepsilon} \subseteq F_{a+2\varepsilon}$.

$$F_a = f^{-1}(-\infty, a]$$

$$G_a = g^{-1}(-\infty, a]$$

The inclusion maps a component of F_a into a component of $G_{a+\varepsilon}$, and vice versa. Call these maps α^ε and β^ε.



Stability

Stability Theorem: $\mathrm{d_I}(\mathrm{T}_f, \mathrm{T}_g) \leq \|f - g\|_\infty$.

Let $\varepsilon = \|f - g\|_\infty$.

$F_a \subseteq G_{a+\varepsilon} \subseteq F_{a+2\varepsilon}$.

$$F_a = f^{-1}(-\infty, a]$$

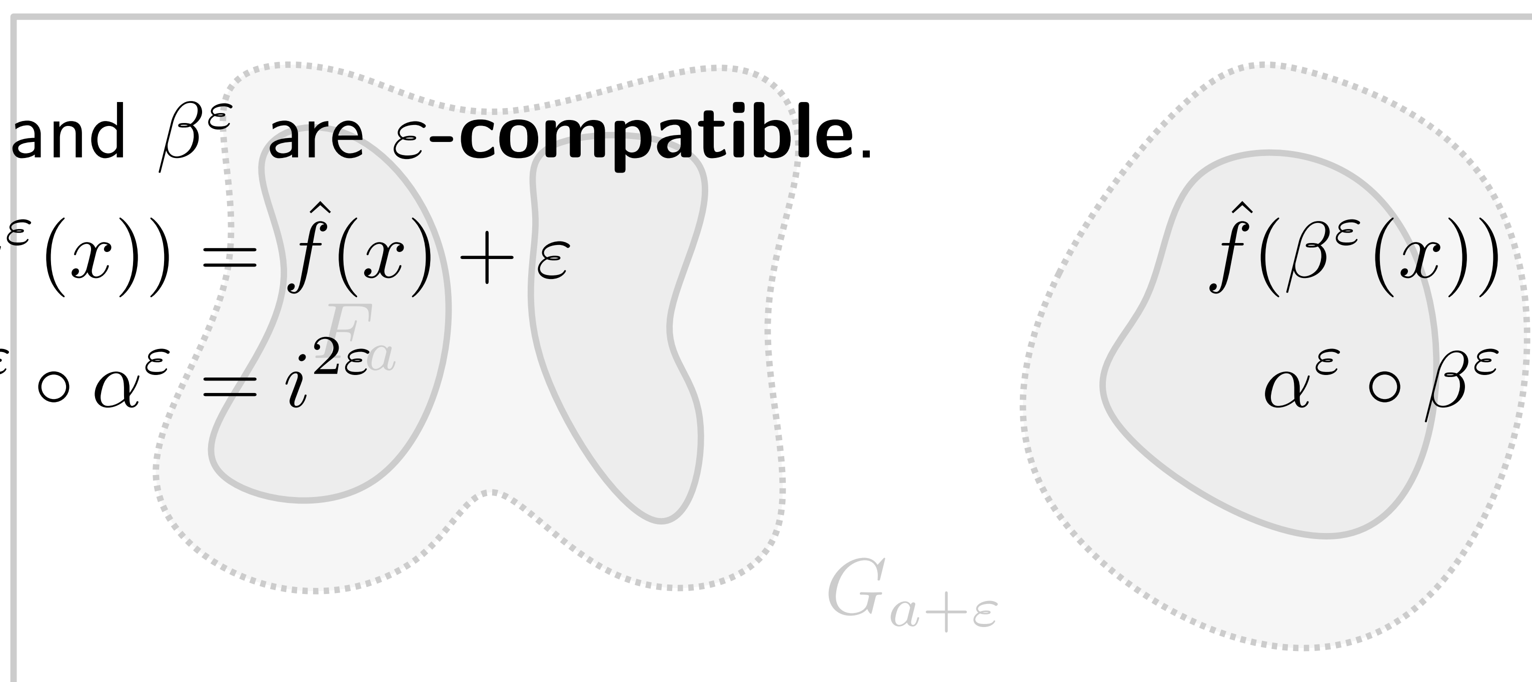
$$G_a = g^{-1}(-\infty, a]$$

The inclusion maps a component of F_a into a component of $G_{a+\varepsilon}$, and vice versa. Call these maps α^ε and β^ε.

Claim: α^ε and β^ε are ε-**compatible**.

$$\hat{g}(\alpha^\varepsilon(x)) = \hat{f}(x) + \varepsilon \qquad \hat{f}(\beta^\varepsilon(x)) = \hat{g}(x) + \varepsilon$$

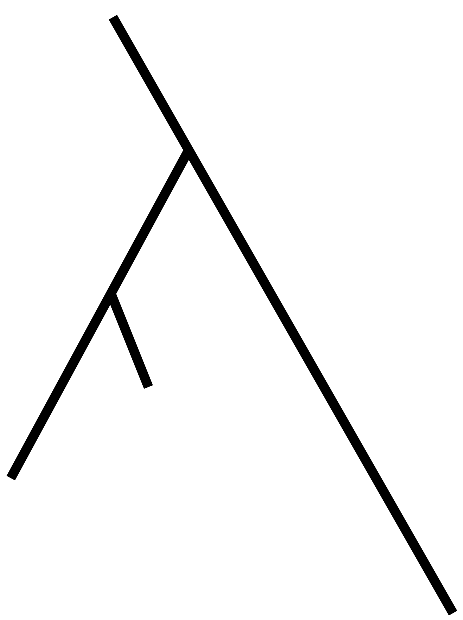
$$\beta^\varepsilon \circ \alpha^\varepsilon = i^{2\varepsilon} \qquad \alpha^\varepsilon \circ \beta^\varepsilon = j^{2\varepsilon}$$

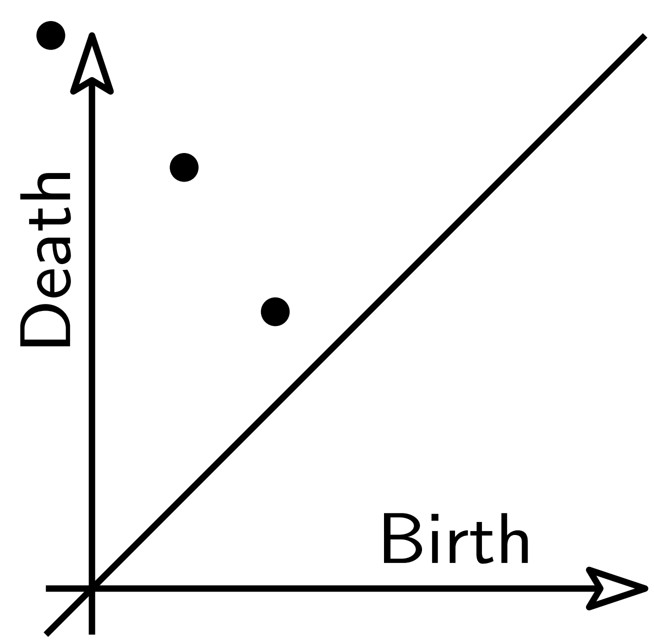


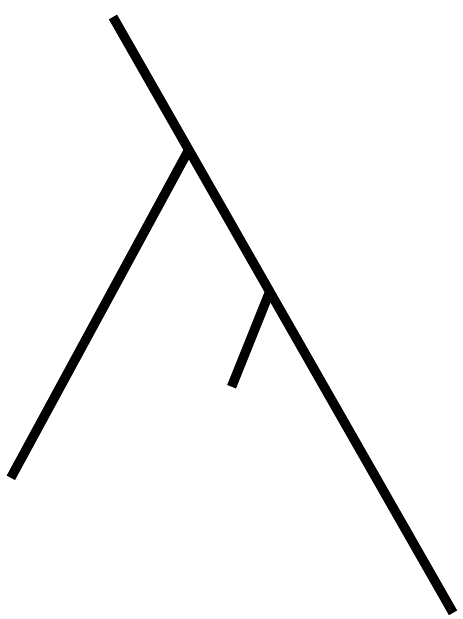
Bottleneck vs. Interleaving Distance

$\mathrm{d_B}(\mathrm{Dgm}_f, \mathrm{Dgm}_g) = 0$

$\mathrm{d_I}(\mathrm{T}_f, \mathrm{T}_g) = \varepsilon > 0$





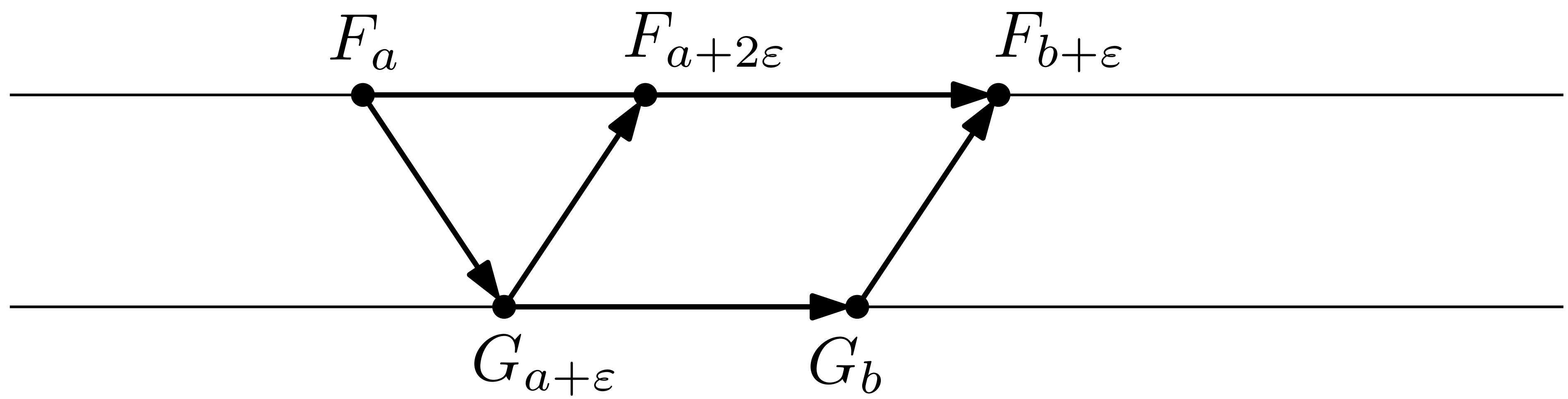


Bottleneck vs. Interleaving Distance

Persistence modules: $\{F_a, i_a^b : F_a \to F_b\}, \{G_a, j_a^b : G_a \to G_b\}$.

ε-**interleaved:**

if there are maps $\phi^a : F_a \to G_{a+\varepsilon}$ and $\psi^a : G_a \to G_{a+\varepsilon}$, such that their compositions commute with i_a^b and j_a^b.



Stability Theorem [Chazal, Cohen-Steiner, Glisse, Guibas, Oudot]:

If two persistence modules are ε-interleaved, then their persistence diagrams are ε-close in the bottleneck distance.

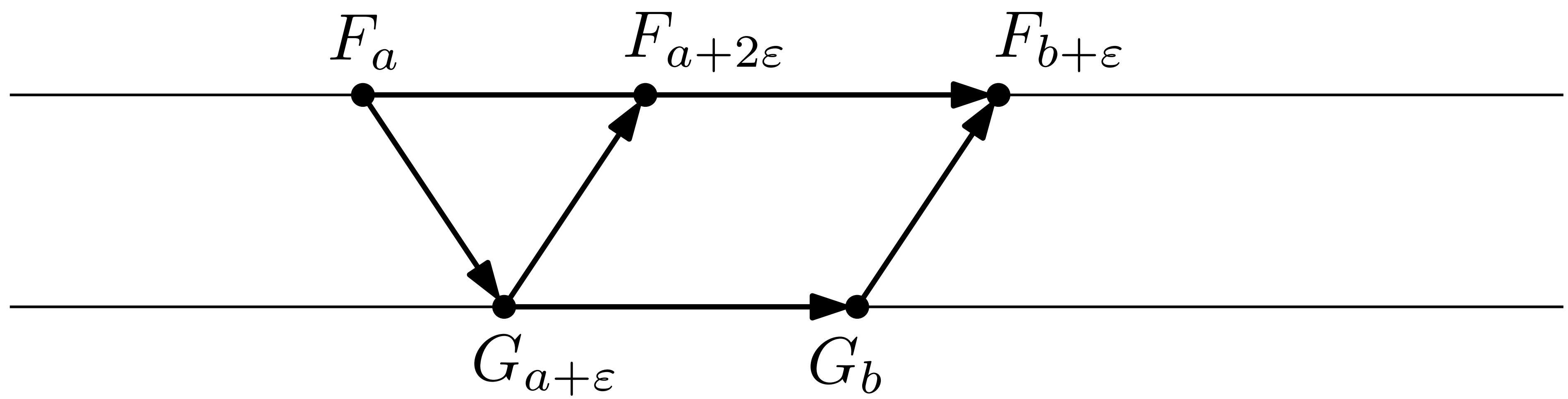
(Generalizes ordinary stability theorem for persistence diagrams if $F_a = \mathsf{H}(f^{-1}(-\infty, a])$ and $G_a = \mathsf{H}(g^{-1}(-\infty, a])$.)

Bottleneck vs. Interleaving Distance

Persistence modules: $\{F_a, i_a^b : F_a \to F_b\}, \{G_a, j_a^b : G_a \to G_b\}$.

ε-**interleaved:**

if there are maps $\phi^a : F_a \to G_{a+\varepsilon}$ and $\psi^a : G_a \to G_{a+\varepsilon}$,
such that their compositions commute with i_a^b and j_a^b.



Stability Theorem [Chazal, Cohen-Steiner, Glisse, Guibas, Oudot]:
If two persistence modules are ε-interleaved, then their persistence diagrams are ε-close in the bottleneck distance.

(Generalizes ordinary stability theorem for persistence diagrams if $F_a = \mathsf{H}(f^{-1}(-\infty, a])$ and $G_a = \mathsf{H}(g^{-1}(-\infty, a])$.)

Corollary: $\mathrm{d_B}(\mathrm{Dgm}_0(f), \mathrm{Dgm}_0(g)) \leq \mathrm{d_I}(f, g)$.

Proof: $\alpha^\varepsilon : T_f \to T_g \Rightarrow \phi^a : \mathsf{H}_0(f^{-1}(-\infty, a]) \to \mathsf{H}_0(g^{-1}(-\infty, a+\varepsilon])$

$\beta^\varepsilon : T_g \to T_f \Rightarrow \psi^a : \mathsf{H}_0(g^{-1}(-\infty, a]) \to \mathsf{H}_0(f^{-1}(-\infty, a+\varepsilon])$

Parallel Computation

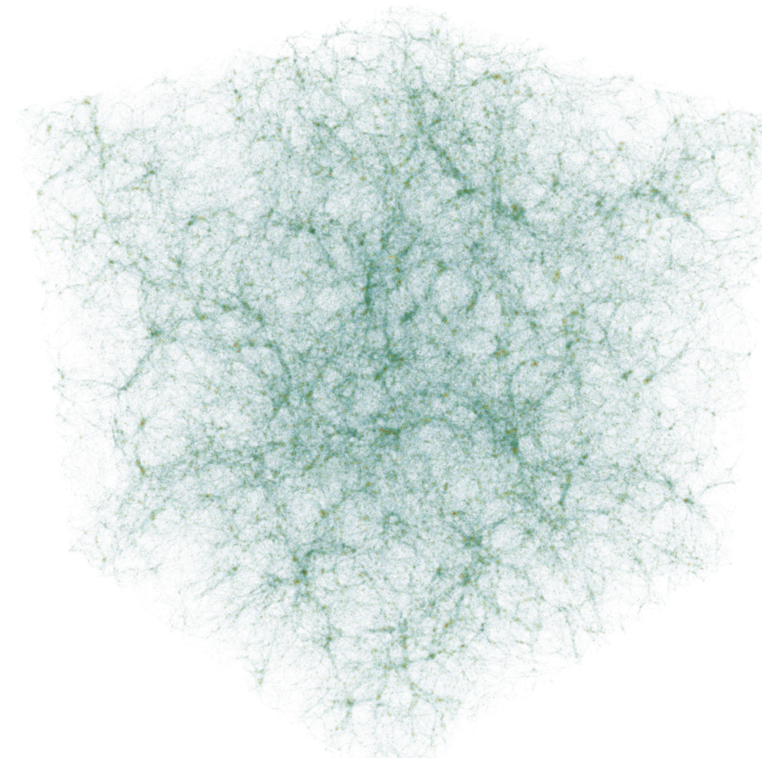
of Merge Trees

Sample Queries

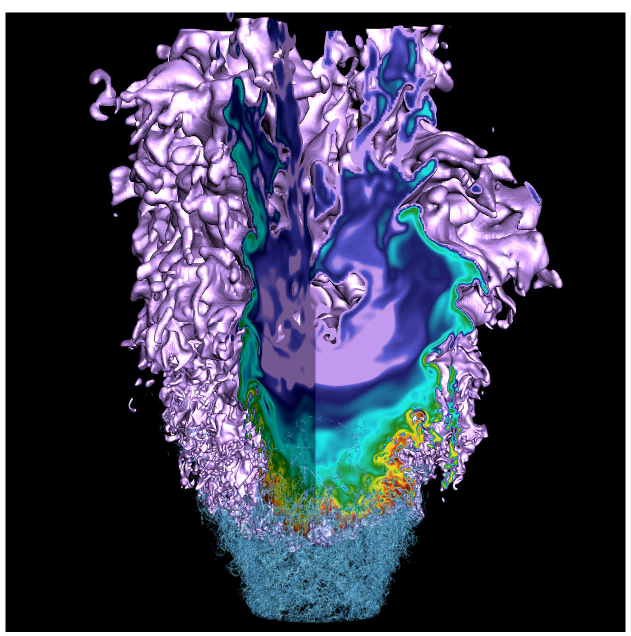
- Cosmological simulations of the universe.

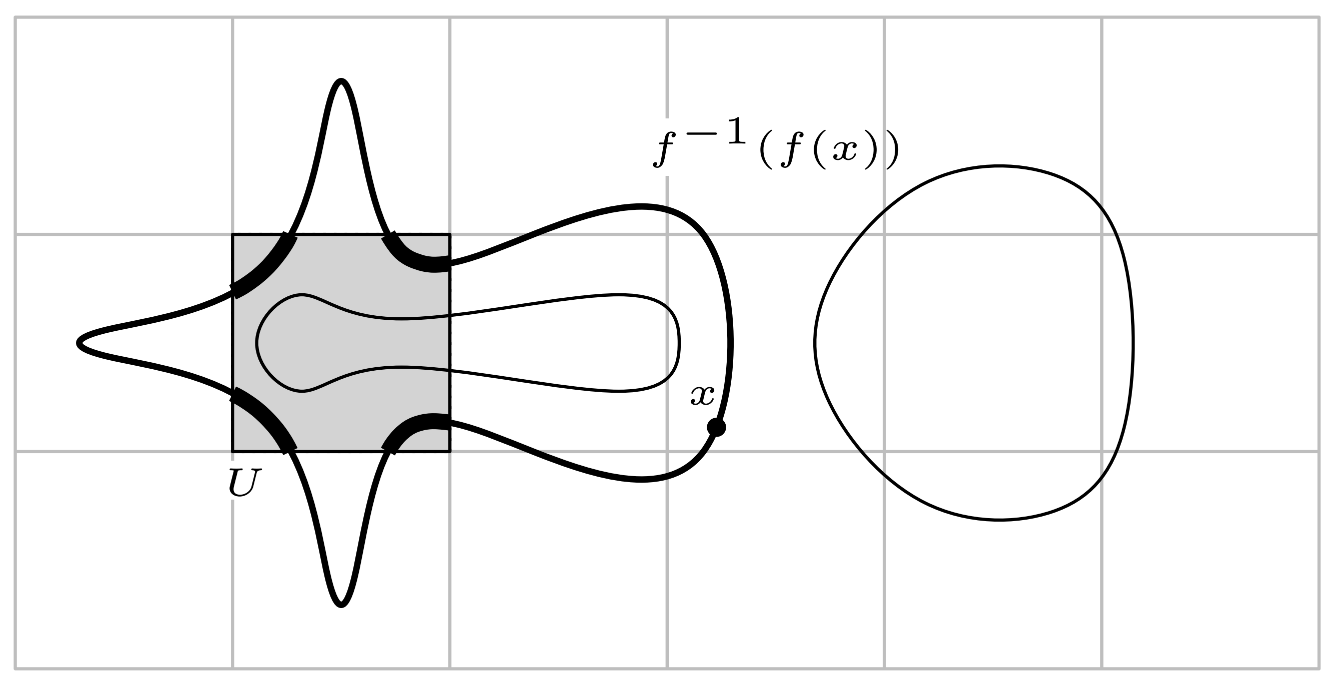
  Compare statistical properties to observations, distribution of mass of heavy objects.

  Detect heavy objects as persistent maxima, but how to integrate their mass in parallel?



- Extract a component of the levelset that contains a specific point.
  (E.g., when studying the consumption of hydrogen during combustion.)





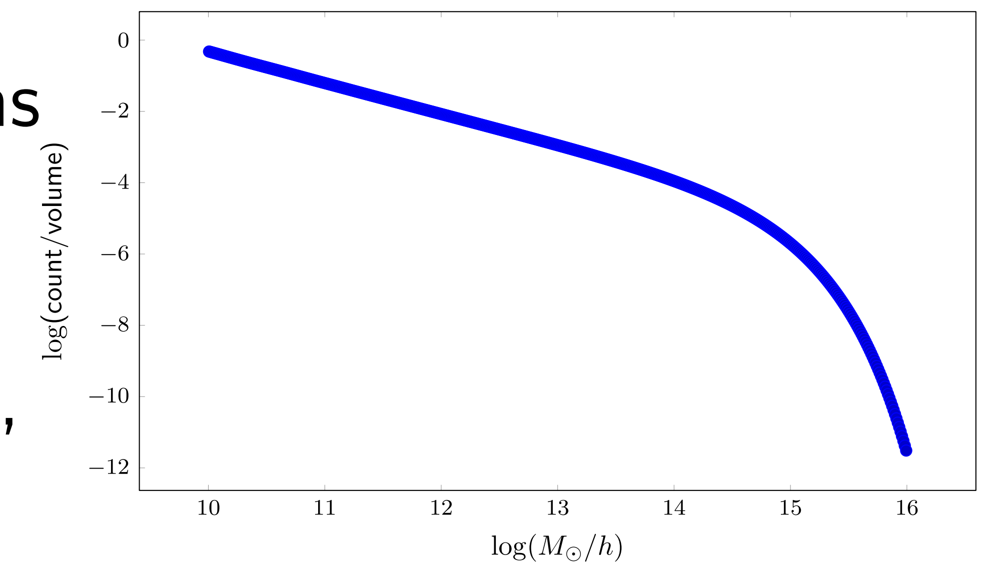
- The datasets are large: $1,024^3 - 4,096^3$ per timestep.

Sample Queries

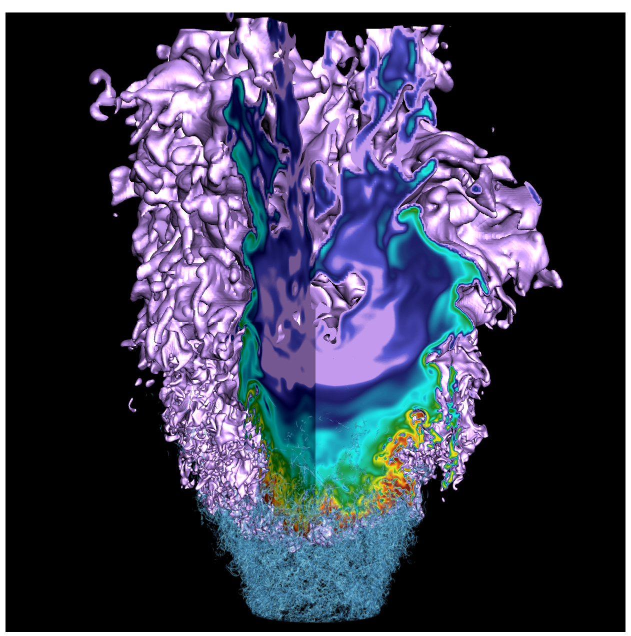
- Cosmological simulations of the universe.

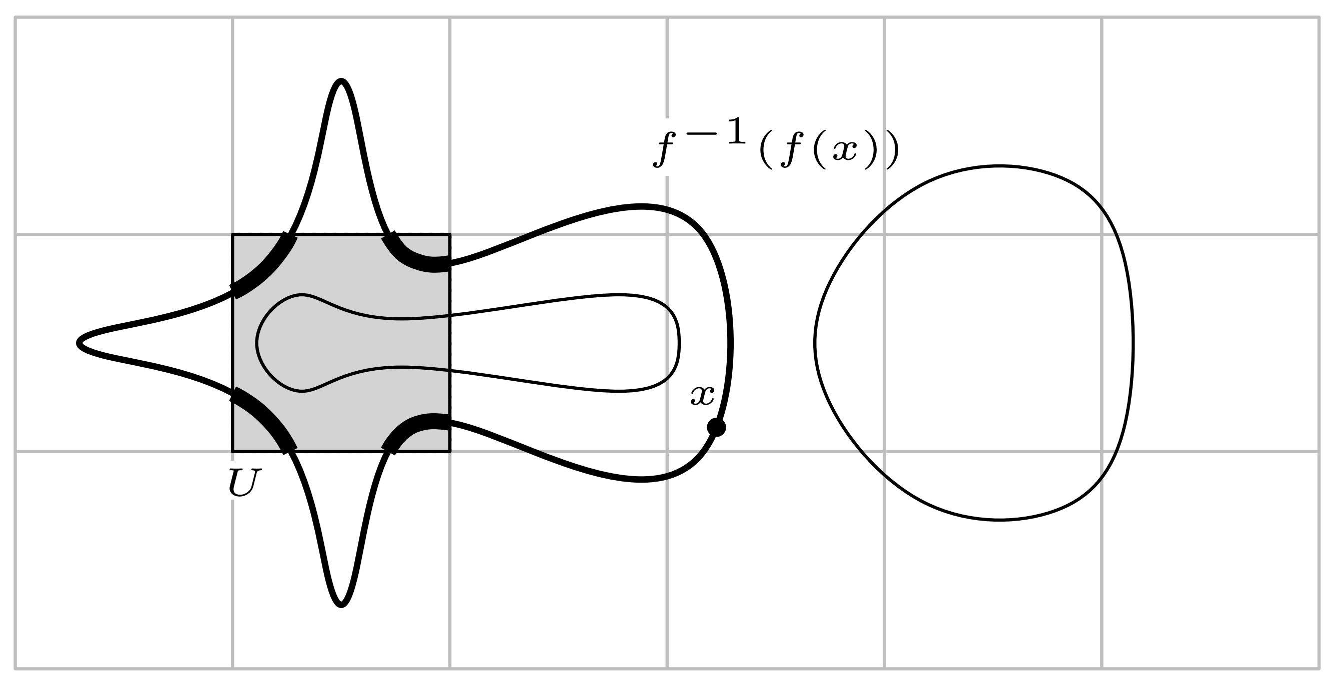
  Compare statistical properties to observations distribution of mass of heavy objects.

  Detect heavy objects as persistent maxima, but how to integrate their mass in parallel?



- Extract a component of the levelset that contains a specific point.
  (E.g., when studying the consumption of hydrogen during combustion.)

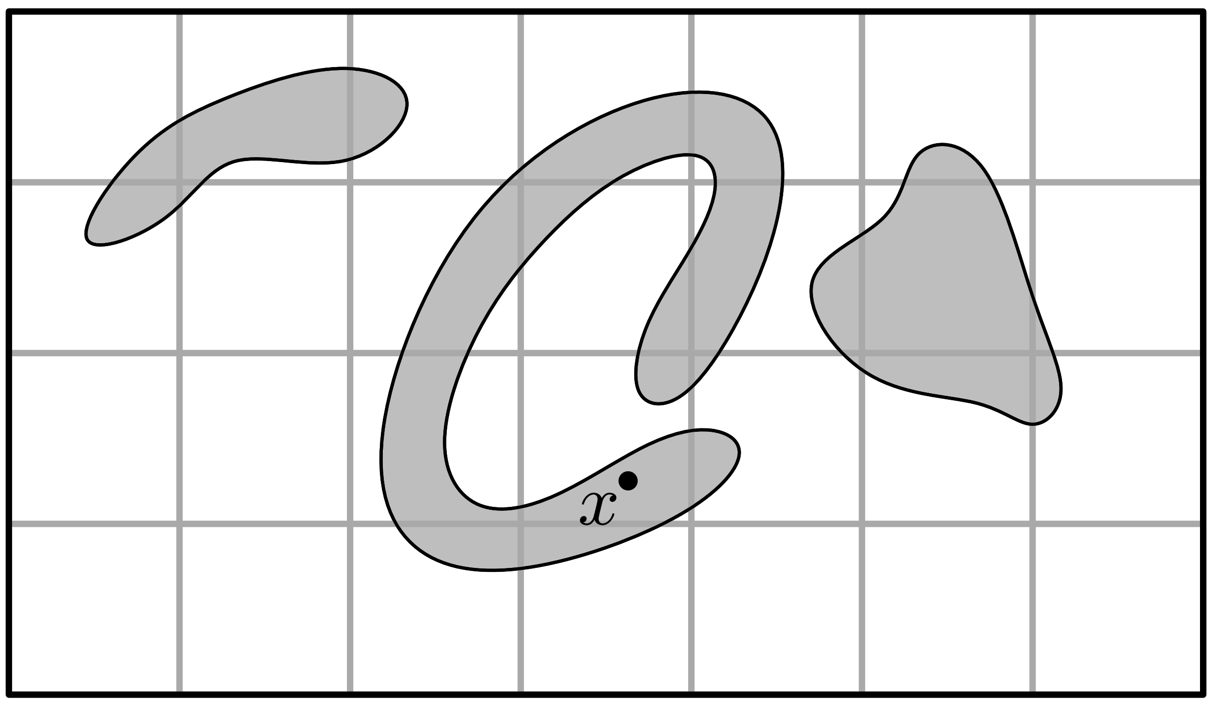




- The datasets are large: $1,024^3 - 4,096^3$ per timestep.

Component volume query

Given a point $x \in \mathbb{X}$, find the volume of the component of the sublevel set $f^{-1}(-\infty, a]$ that contains x.



(e.g., determine the volume of a cluster)

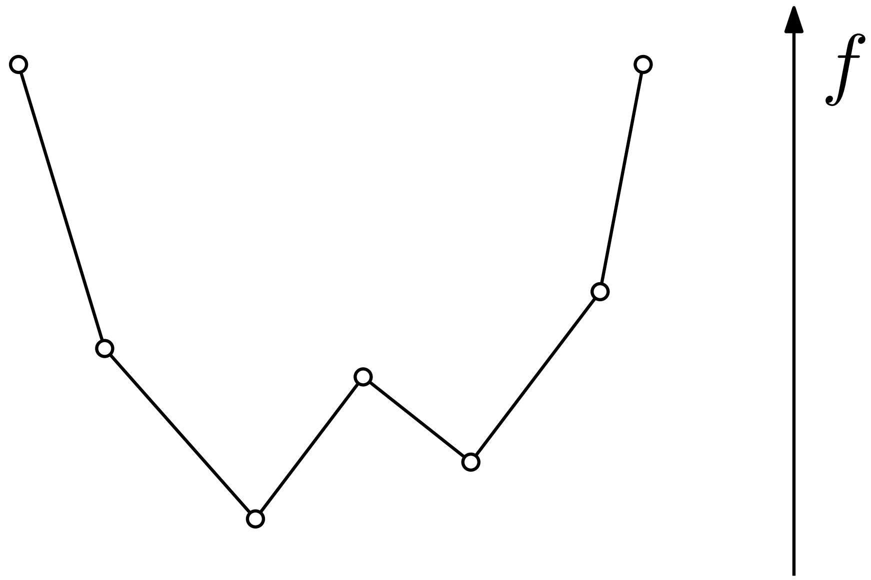
- Brute-force solution is too slow when the data is distributed among many processors;
- It makes even less sense if one is interested in a histogram of volumes as we vary the sublevelset thresholds.

Merge Trees: Construction

Function: $f : K \to \mathbb{R}$

K is a triangulation;

f is defined on the vertices and piecewise-linearly interpolated.



Merge Trees: Construction

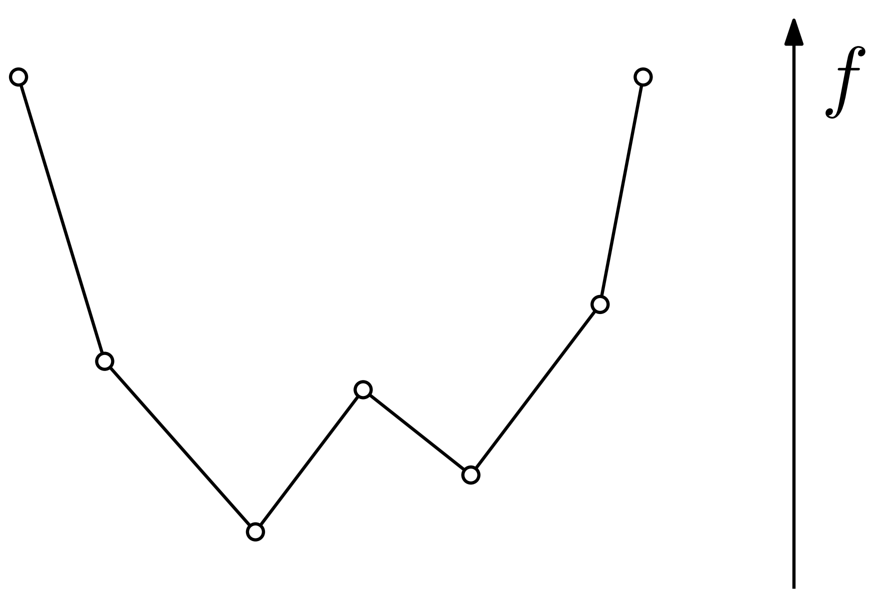
Function: $f : K \to \mathbb{R}$

K is a triangulation;

f is defined on the vertices and piecewise-linearly interpolated.

Merge tree construction:

sort vertices of K by f
for each vertex v in sorted order **do**
  add v
  for each edge uv with $f(u) \leq f(v)$ **do**
    if $\textsc{Find}(u) \neq \textsc{Find}(v)$ **then**
      set v as the parent of u in T
      $\textsc{Union}(u, v)$



Merge Trees: Construction

Function: $f : K \to \mathbb{R}$

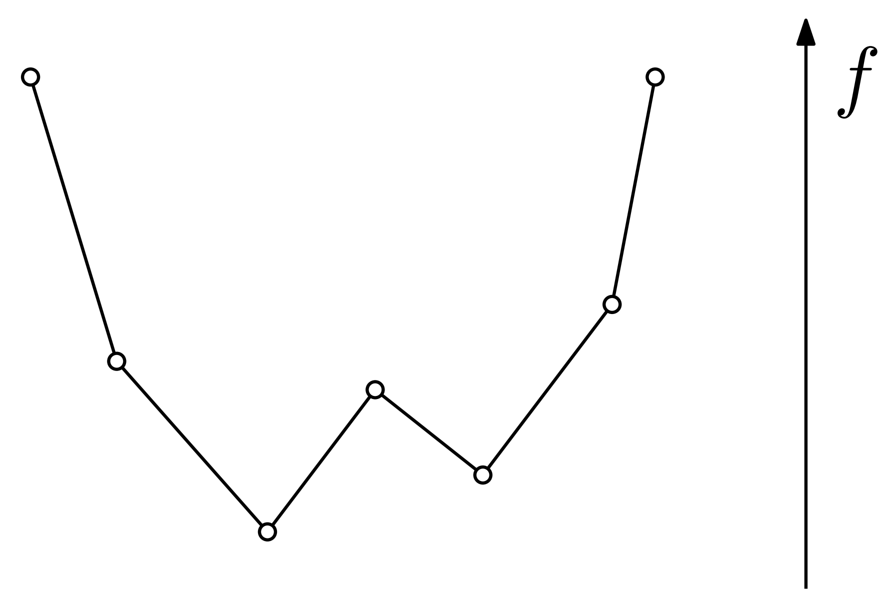
K is a triangulation;

f is defined on the vertices and piecewise-linearly interpolated.

Merge tree construction:

Kruskal's algorithm

sort vertices of K by f
for each vertex v in sorted order **do**
  add v
  for each edge uv with $f(u) \leq f(v)$ **do**
    if $\textsc{Find}(u) \neq \textsc{Find}(v)$ **then**
      set v as the parent of u in T
      $\textsc{Union}(u, v)$



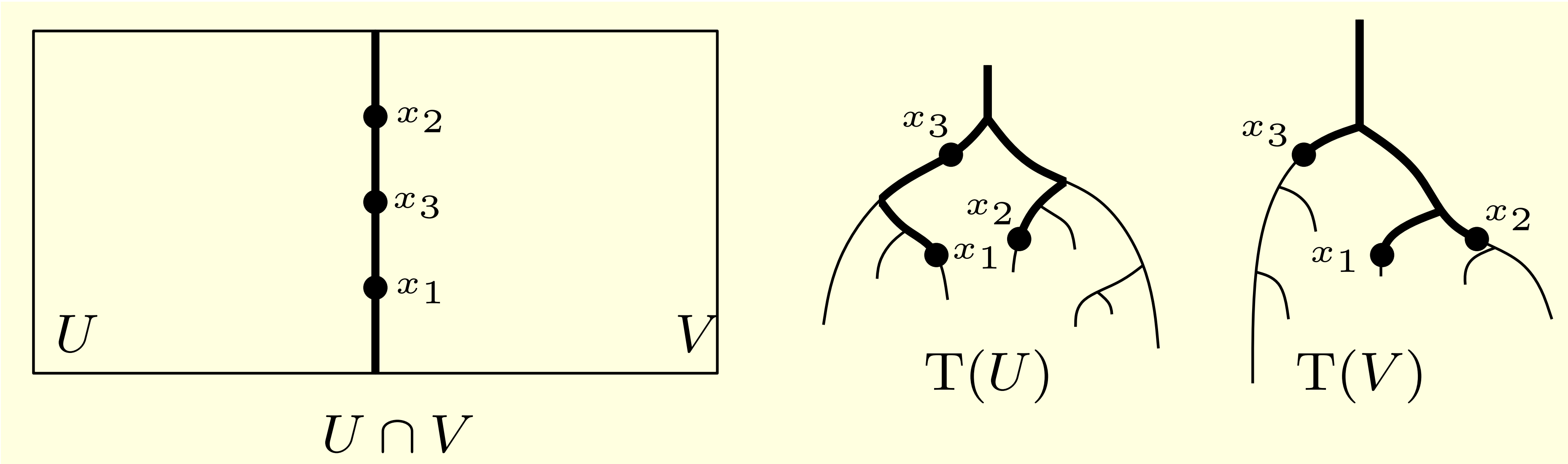
Connection to MST?

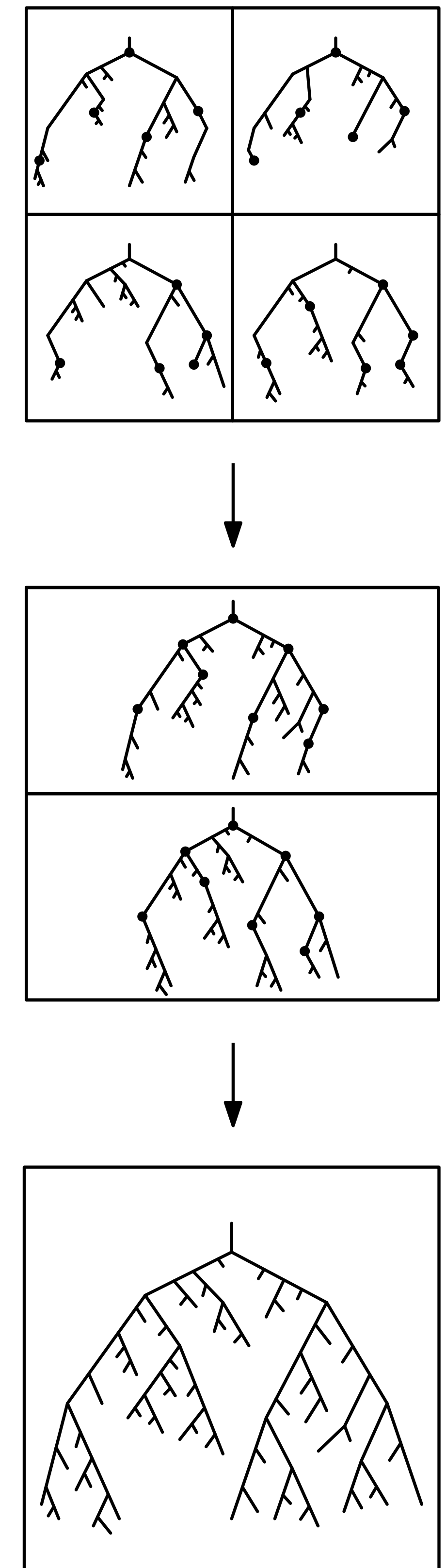
- Best known deterministic algorithm for MST: $\mathrm{O}(m\alpha(m, n))$ [Chazelle '00]
- Lower-bound for merge trees: $\Omega(n \log n)$.

$n = |\text{vertices}|$

$m = |\text{edges}|$

Existing Parallel Approach [Pascucci, Cole-McLaughlin '03]





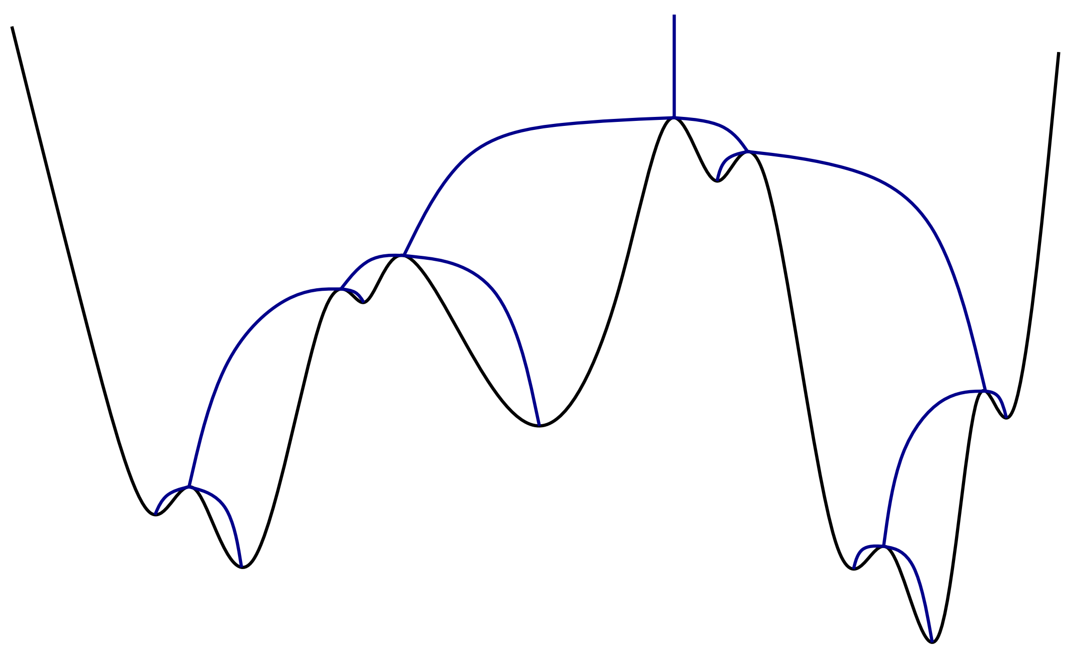
- Hierarchically partition the domain (e.g., a quad- or oct-tree for regular grids).
- On each processor P_i, compute the merge tree T_{U_i} of the function restricted to the set U_i.
- **Merge trees in pairs**, until we get the full merge tree. (In other words, perform a binary reduction.)

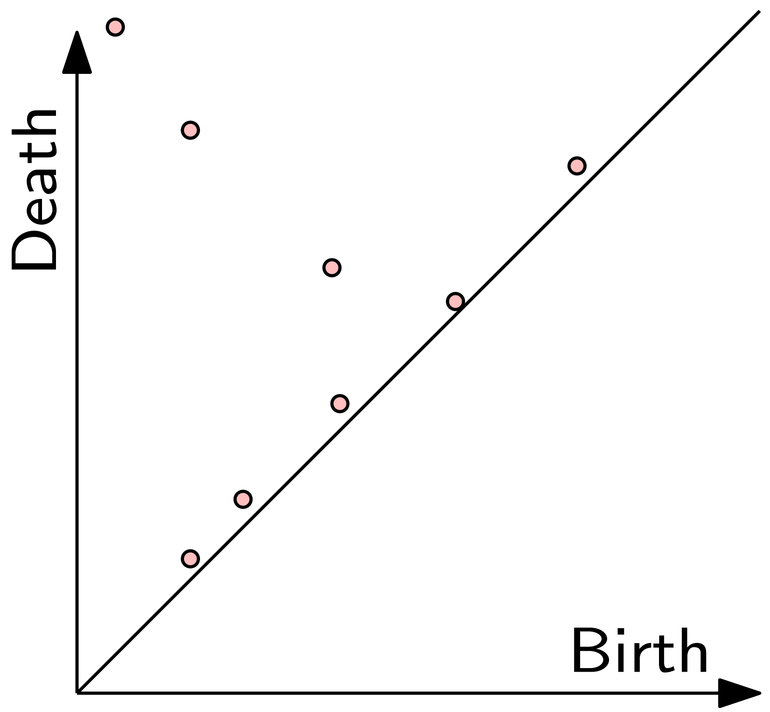
Problem: The reduction is top-heavy. At the end, a single processor has to assemble the entire merge tree. The procedure does not scale.

Solution I: Global Simplified

Data is always corrupted by **noise**.
Typical analysis pipeline: compute a descriptor; simplify the descriptor; use the simplified descriptor for analysis.

For merge trees, simplification means pruning short banches.
Given $\varepsilon > 0$, remove subtrees of depth less than ε.

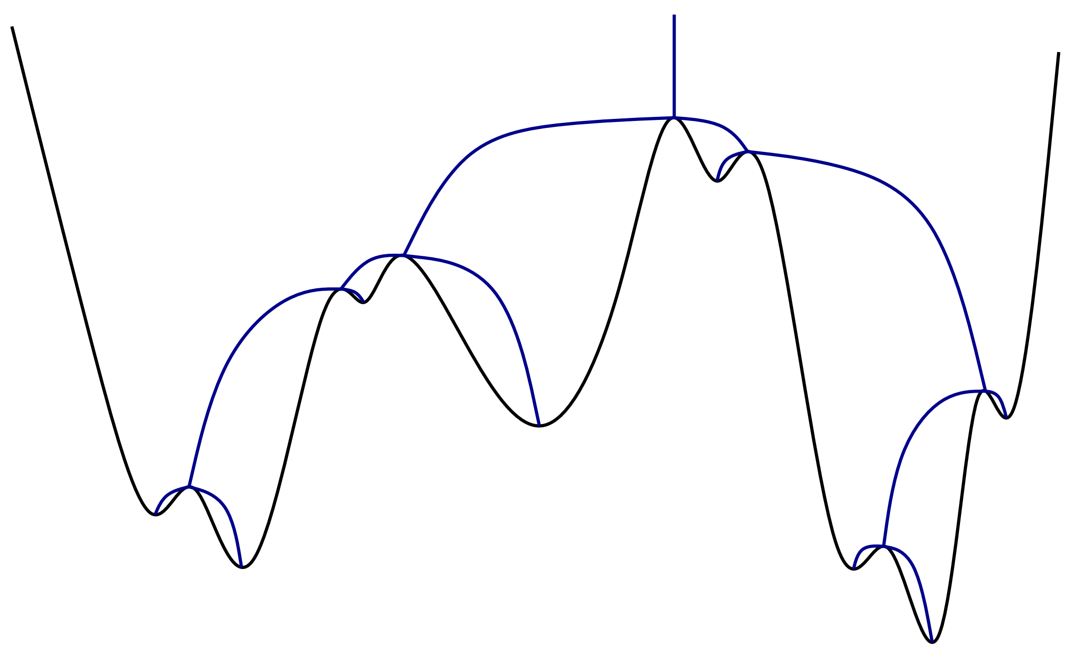


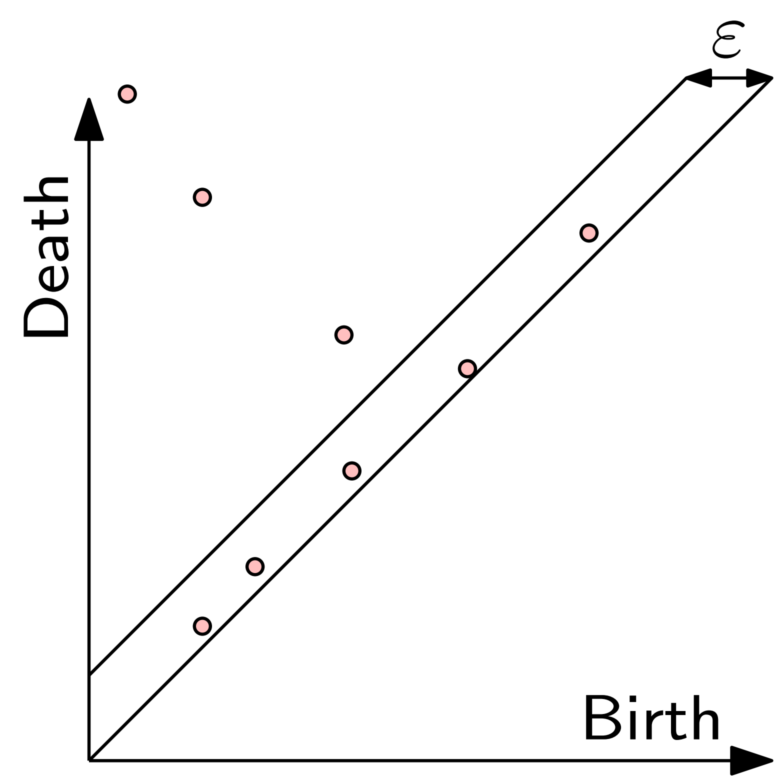


Solution I: Global Simplified

Data is always corrupted by **noise**.
Typical analysis pipeline: compute a descriptor; simplify the descriptor; use the simplified descriptor for analysis.

For merge trees, simplification means pruning short banches.
Given $\varepsilon > 0$, remove subtrees of depth less than ε.

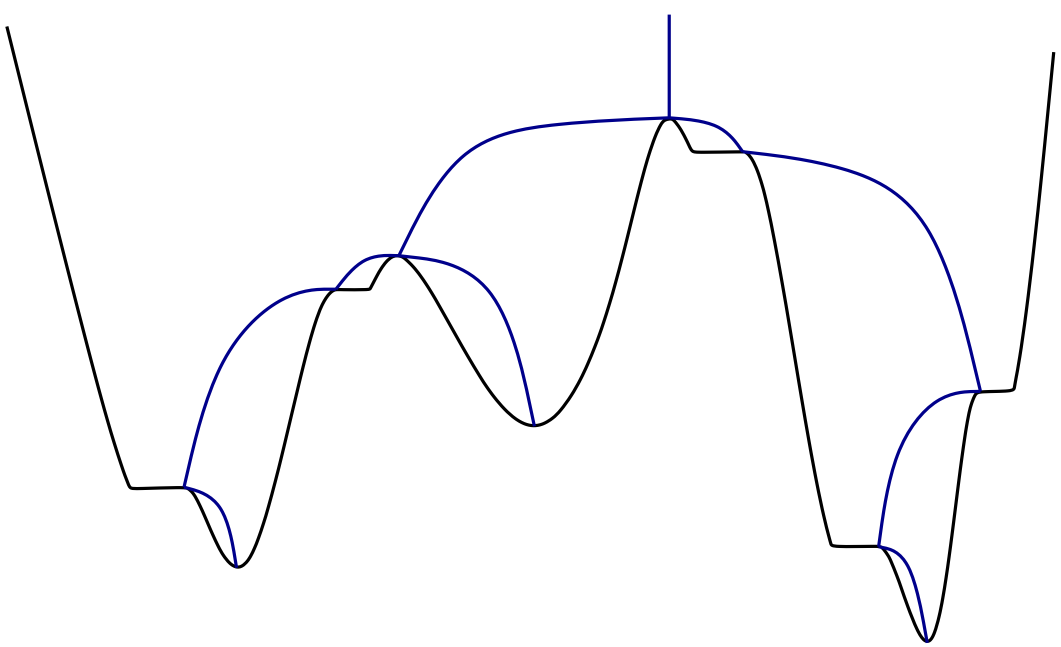


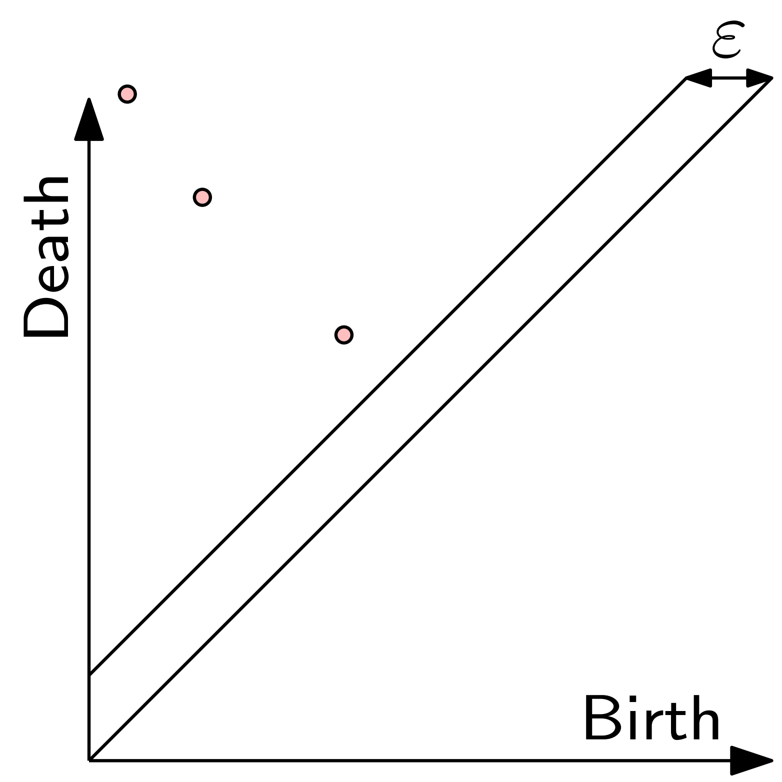


Solution I: Global Simplified

Data is always corrupted by **noise**.
Typical analysis pipeline: compute a descriptor; simplify the descriptor; use the simplified descriptor for analysis.

For merge trees, simplification means pruning short banches.
Given $\varepsilon > 0$, remove subtrees of depth less than ε.





Interpretation:
Given $f : \mathbb{X} \to \mathbb{R}$, there is $g : \mathbb{X} \to \mathbb{R}$, with $\|f - g\|_\infty \leq \varepsilon$, such that g has the fewest extrema. Compute the merge tree of g, rather than f.

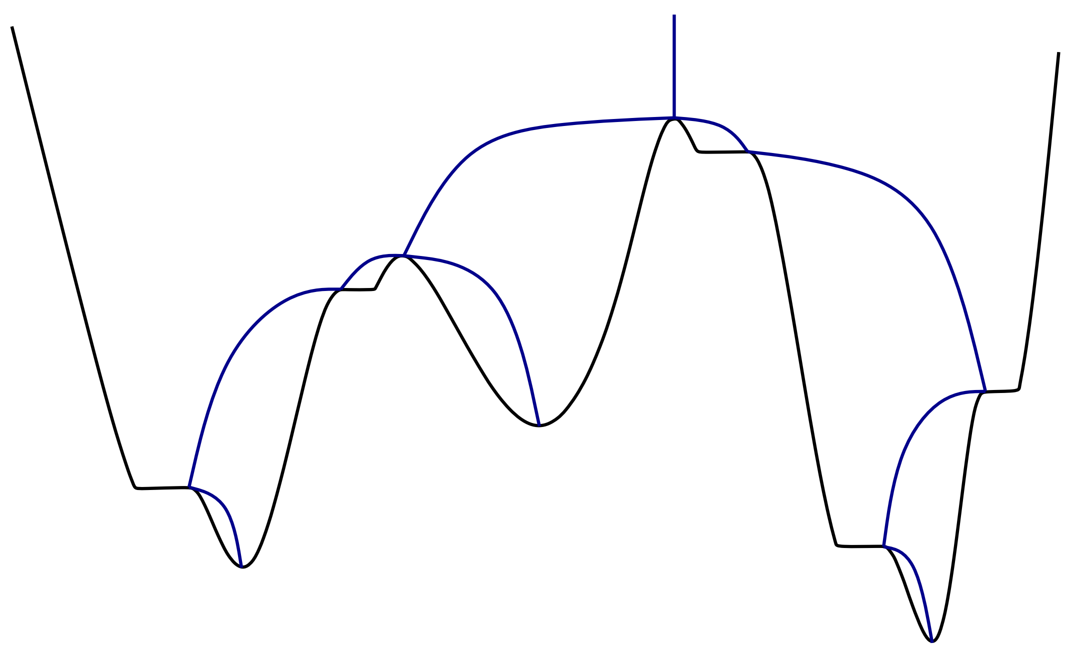
Solution I: Global Simplified

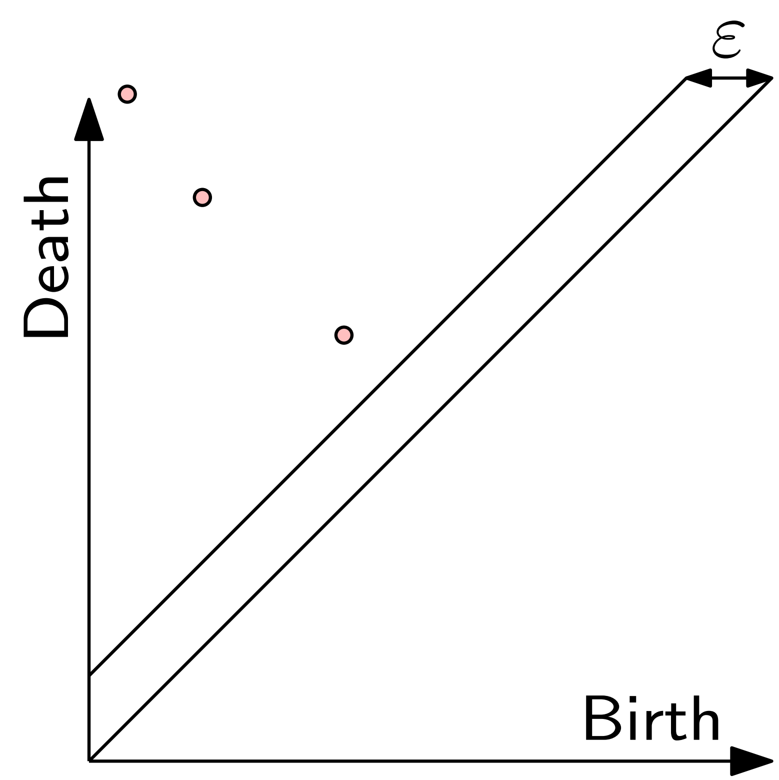
Data is always corrupted by **noise**.

Typical analysis pipeline: compute a descriptor; simplify the descriptor; use the simplified descriptor for analysis.

compute the simplified tree directly

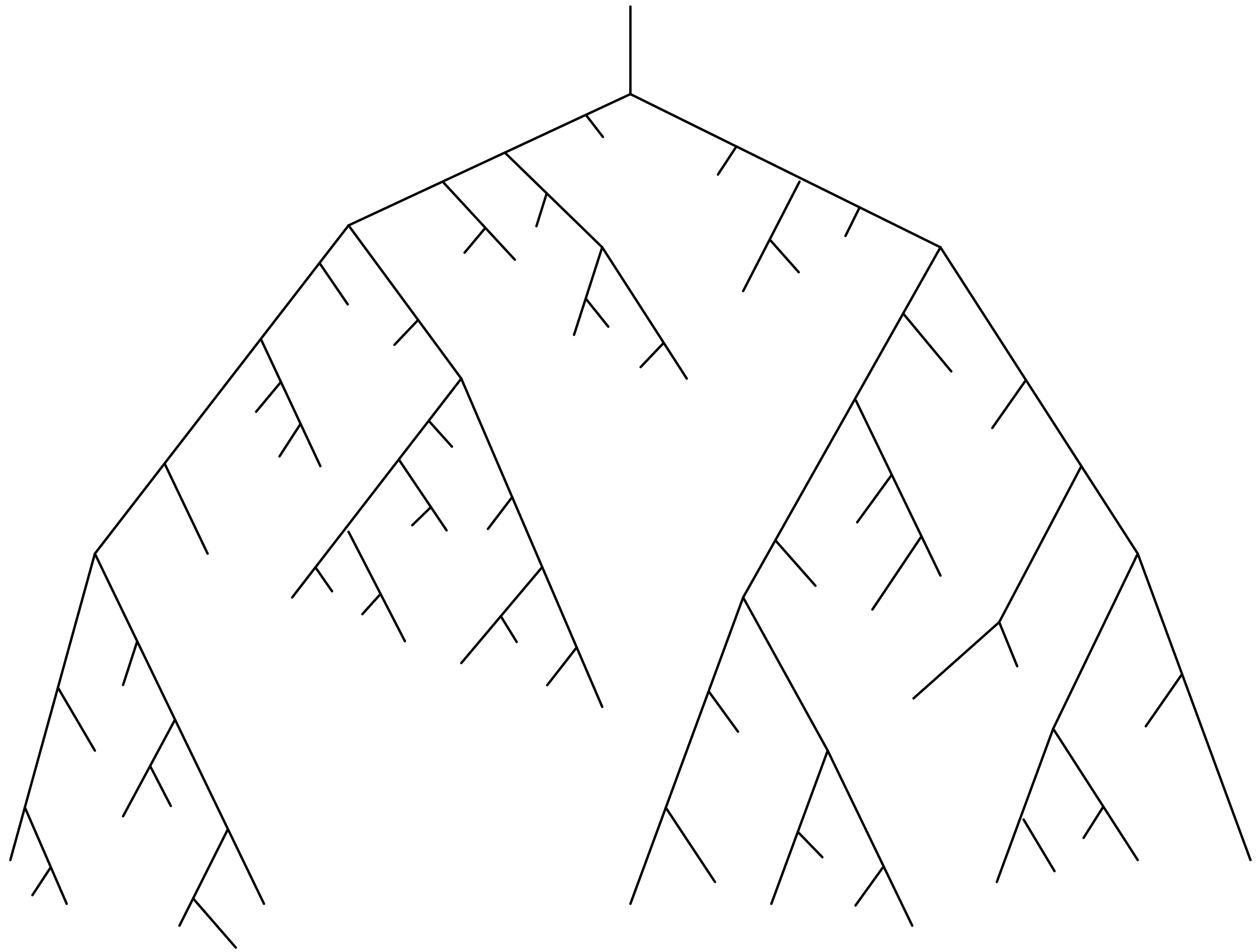
For merge trees, simplification means pruning short banches.
Given $\varepsilon > 0$, remove subtrees of depth less than ε.



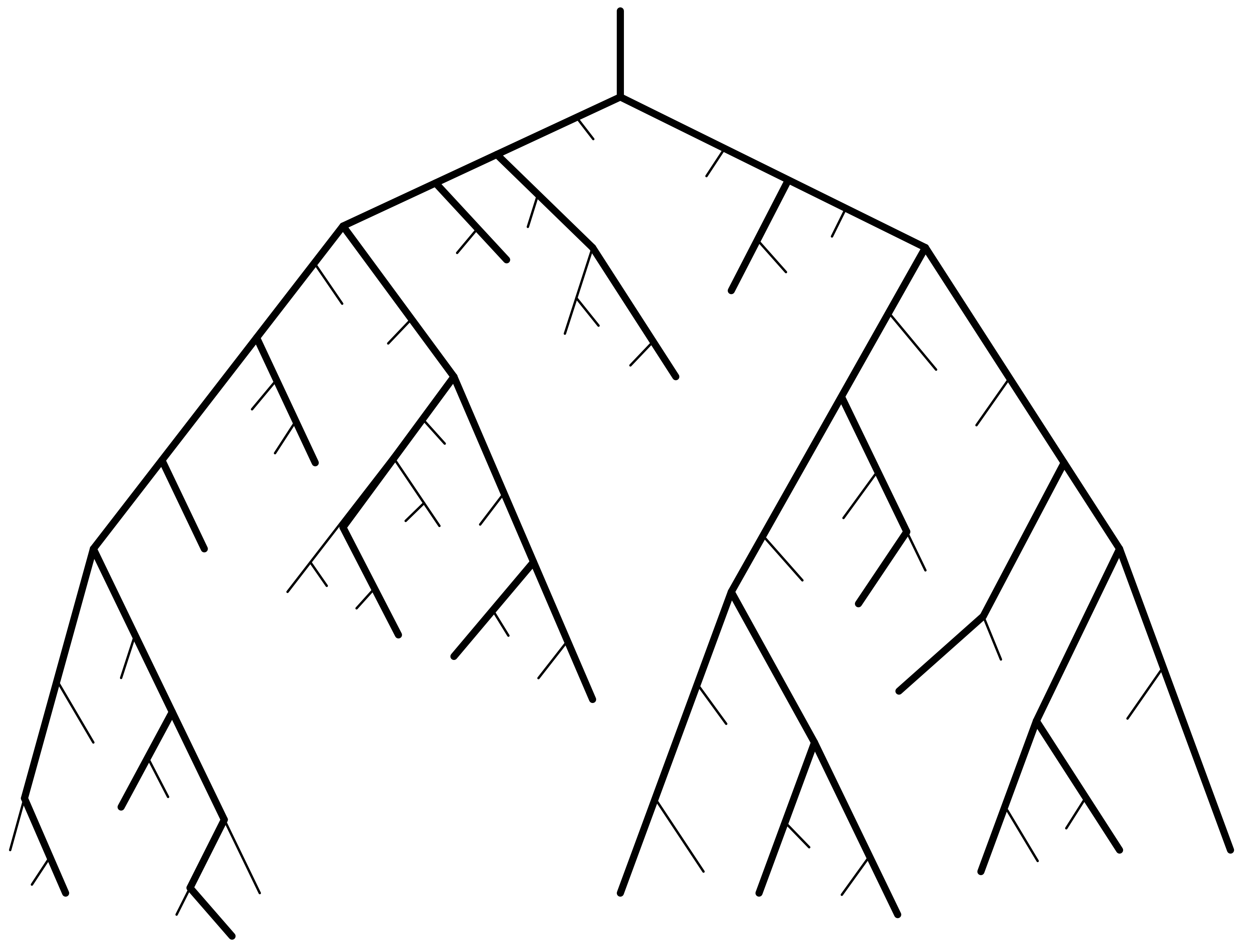


Interpretation:
Given $f : \mathbb{X} \to \mathbb{R}$, there is $g : \mathbb{X} \to \mathbb{R}$, with $\|f - g\|_\infty \leq \varepsilon$, such that g has the fewest extrema. Compute the merge tree of g, rather than f.

Global Simplified



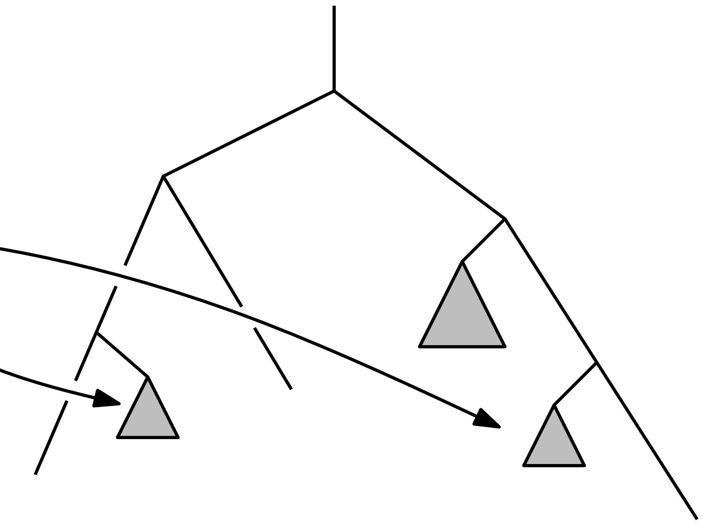
Global Simplified



Interleaved Computation

Theorem: once a subtree lies in the interior of a region, it does not change in the merging process.

low persistence + interior nodes only
$\Rightarrow$ simplify away

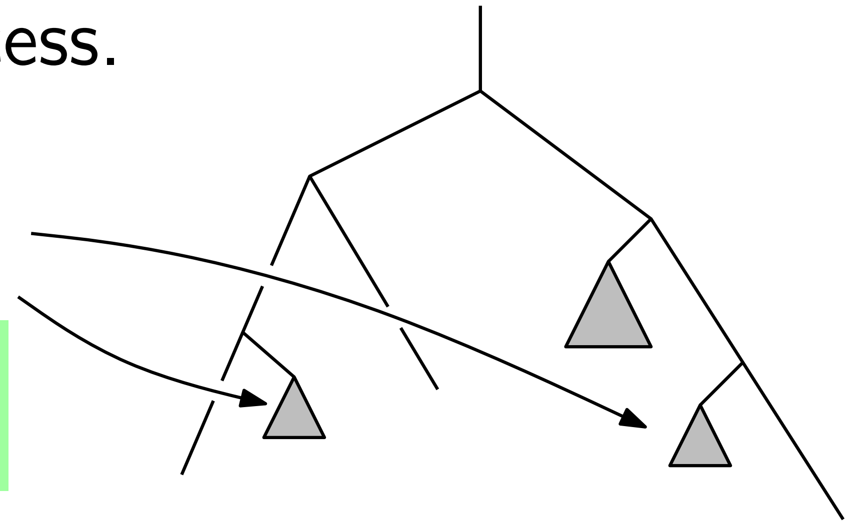


$\Rightarrow$ simplification and merging can be interleaved

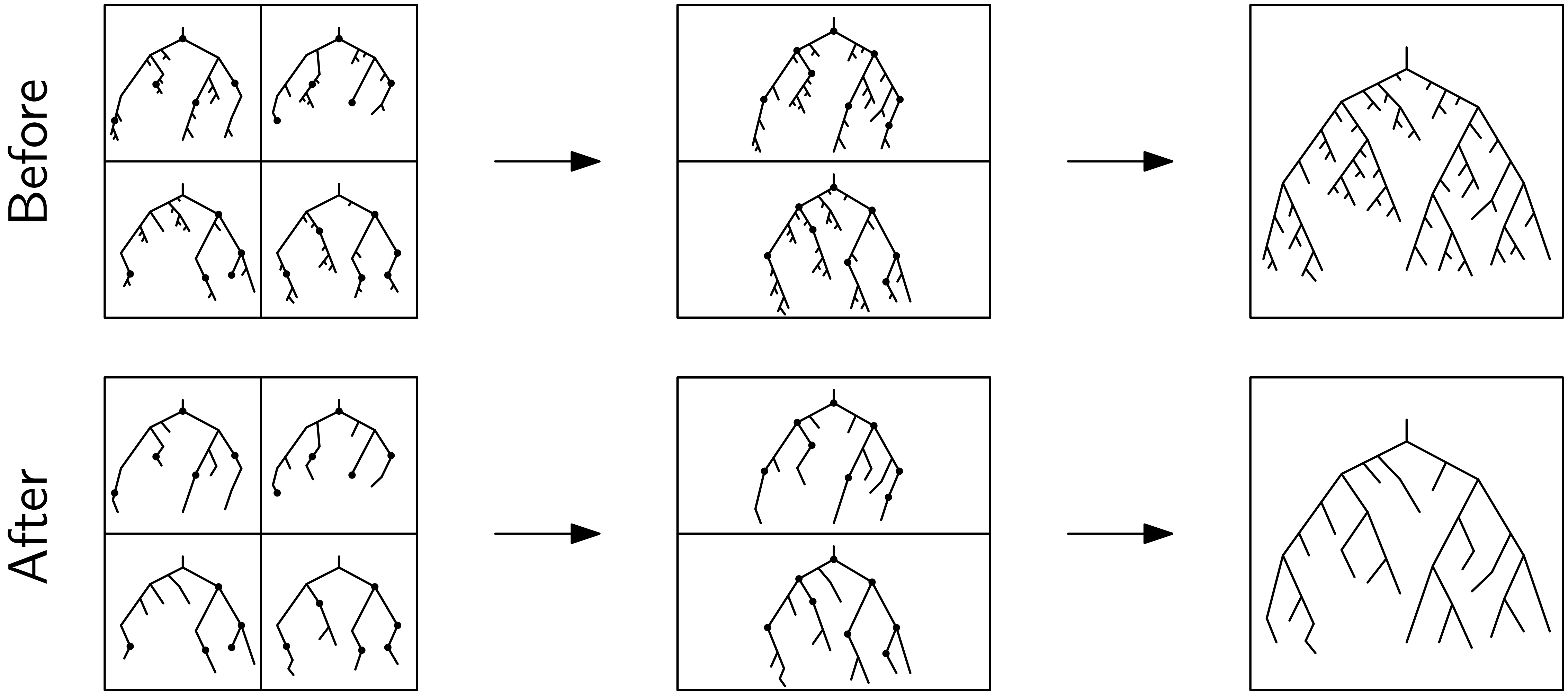
Interleaved Computation

Theorem: once a subtree lies in the interior of a region, it does not change in the merging process.

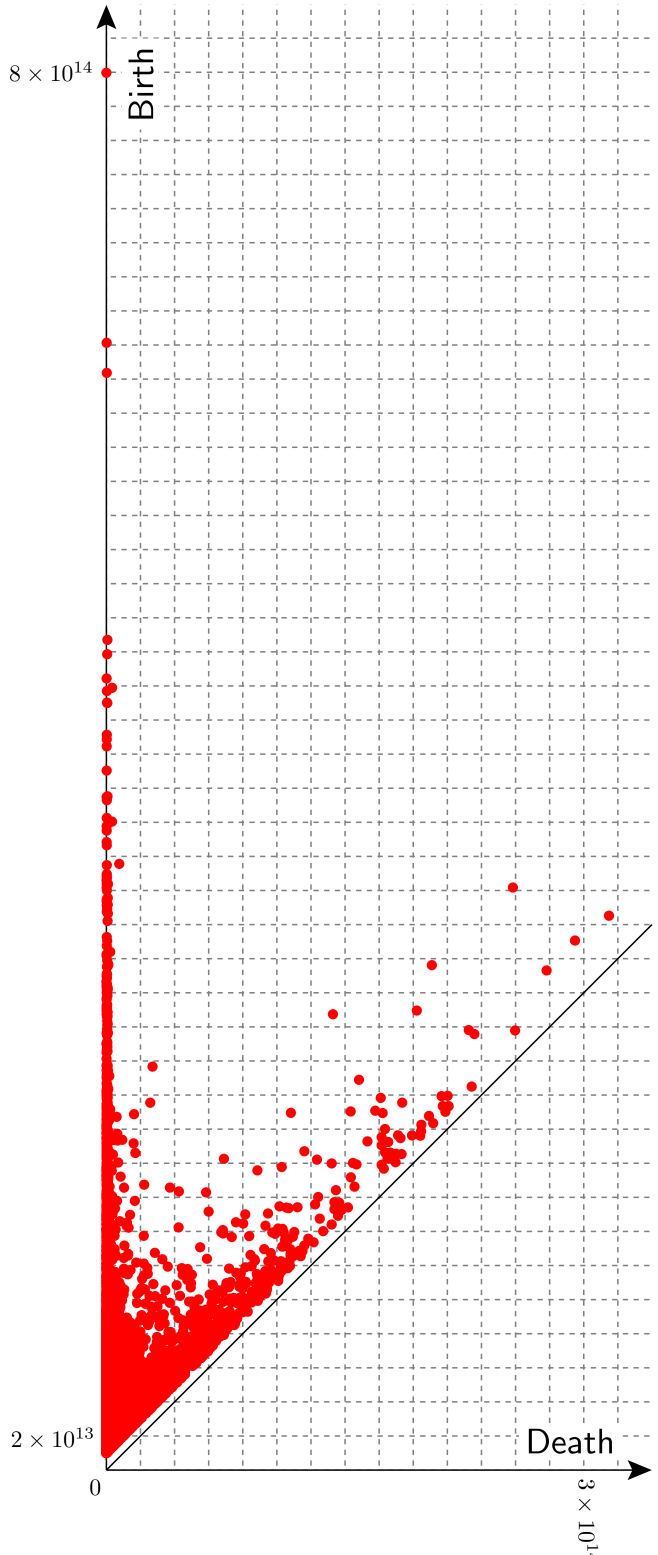
low persistence + interior nodes only
$\Rightarrow$ simplify away

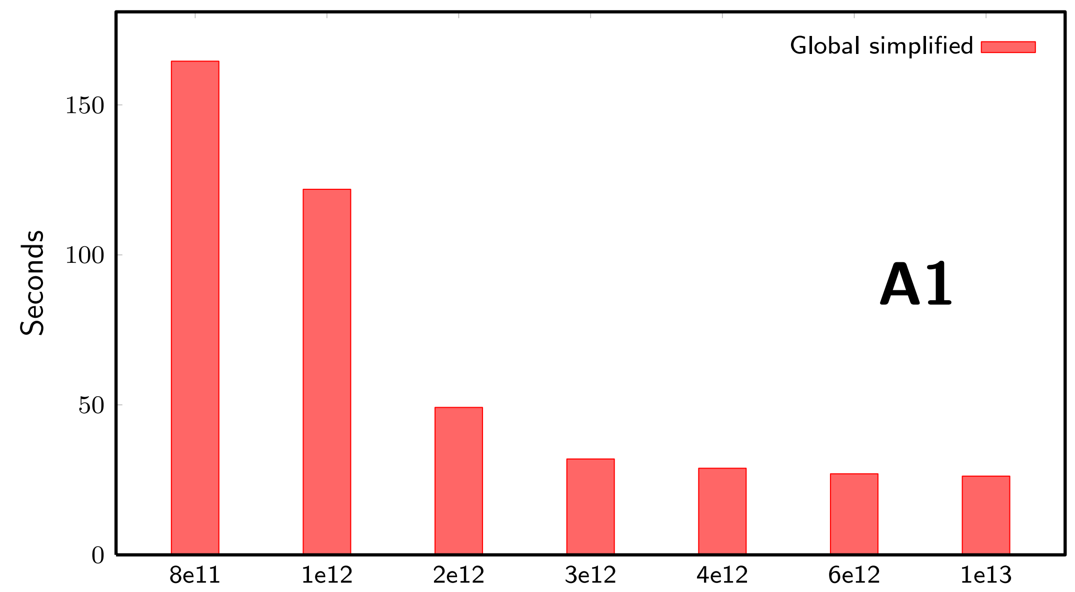


$\Rightarrow$ simplification and merging can be interleaved



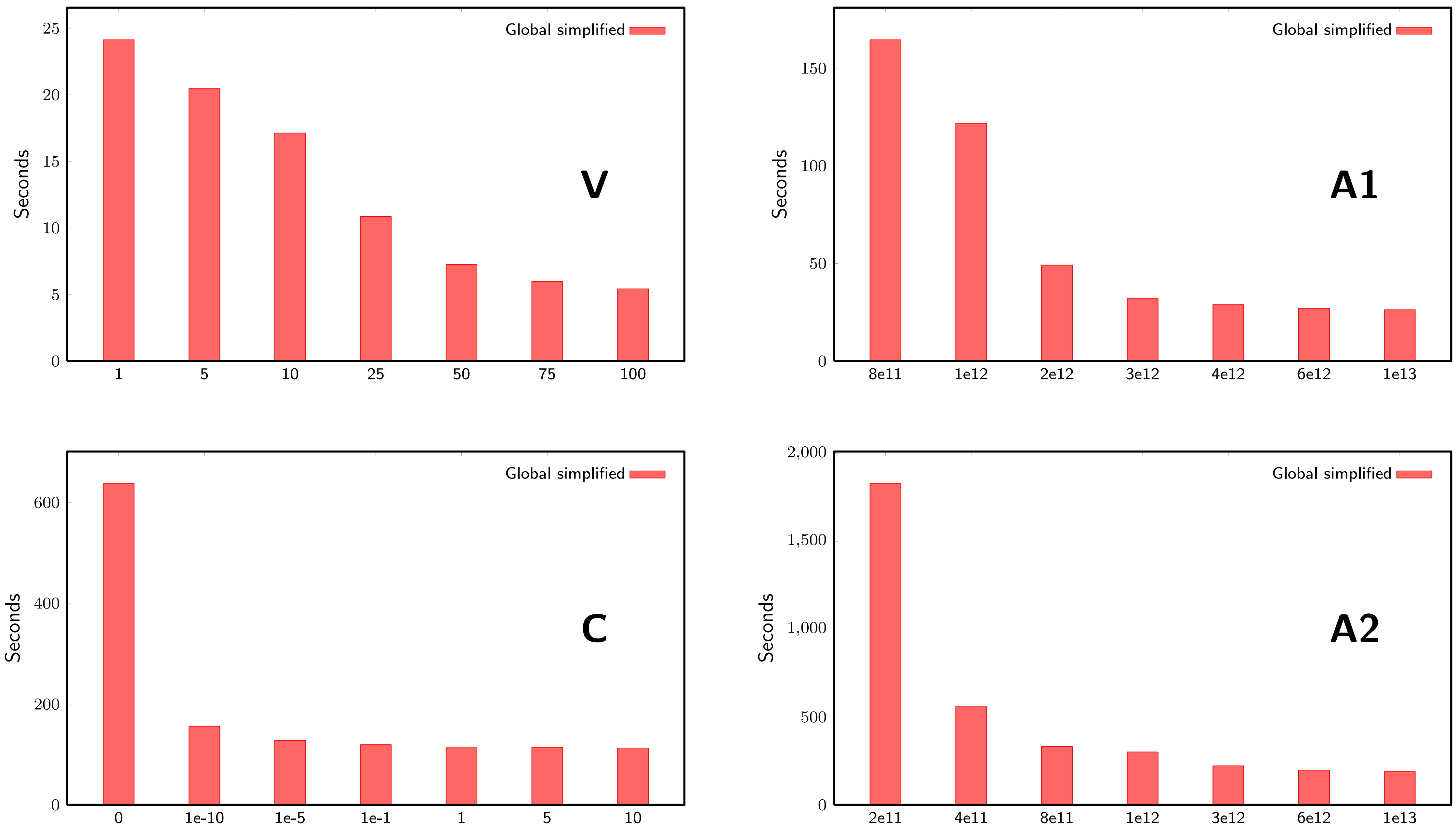
Timings (using 512 processors)





A1 (1024^3): astrophysics simulation

Timings (using 512 processors)



All the experiments performed at the National Energy Research Scientific Computing Center (NERSC) on a Cray XE6 with 24-core AMD 2.1GHz processors per node, sharing 32GB memory.

A2	(2048^3):	astrophysics simulation
C	$(1024^2 \times 2048)$:	combustion simulation
A1	(1024^3):	astrophysics simulation
V	(512^3):	rotational angiography scan

Solution II: Local–Global Representation

Limitations of the global simplified scheme:

- have to pick the simplification threshold ε in advance (chicken-and-egg);
- one monolithic tree in the end (difficult to process).

Solution II: Local–Global Representation

Limitations of the global simplified scheme:

- have to pick the simplification threshold ε in advance (chicken-and-egg);
- one monolithic tree in the end (difficult to process).

Goal: distribute the tree representation.

- Many ways to do this, e.g., could store for every local vertex its parent in the global tree. (Terrible for analysis.)
- **Focus on analysis**: distribute the tree to minimize communication when post-processing.

Solution II: Local–Global Representation

Limitations of the global simplified scheme:

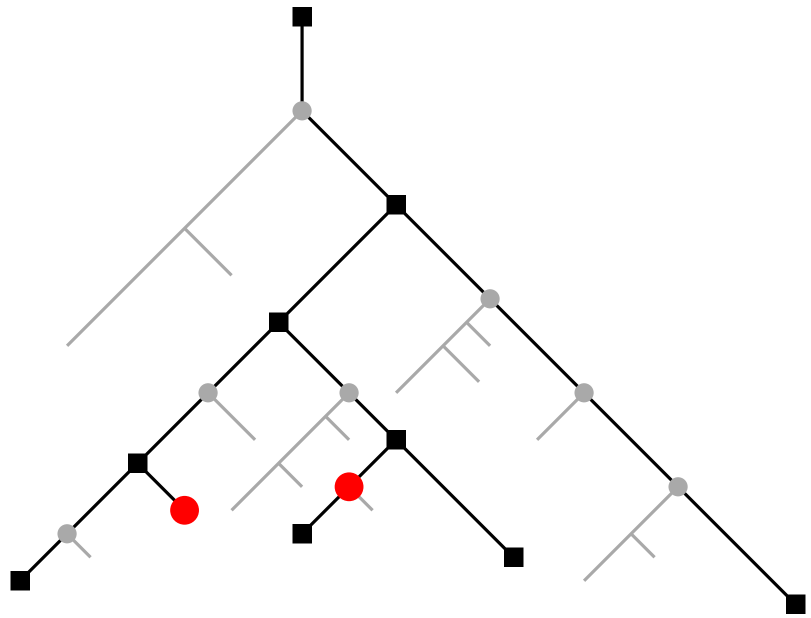
- have to pick the simplification threshold ε in advance (chicken-and-egg);
- one monolithic tree in the end (difficult to process).

Goal: distribute the tree representation.

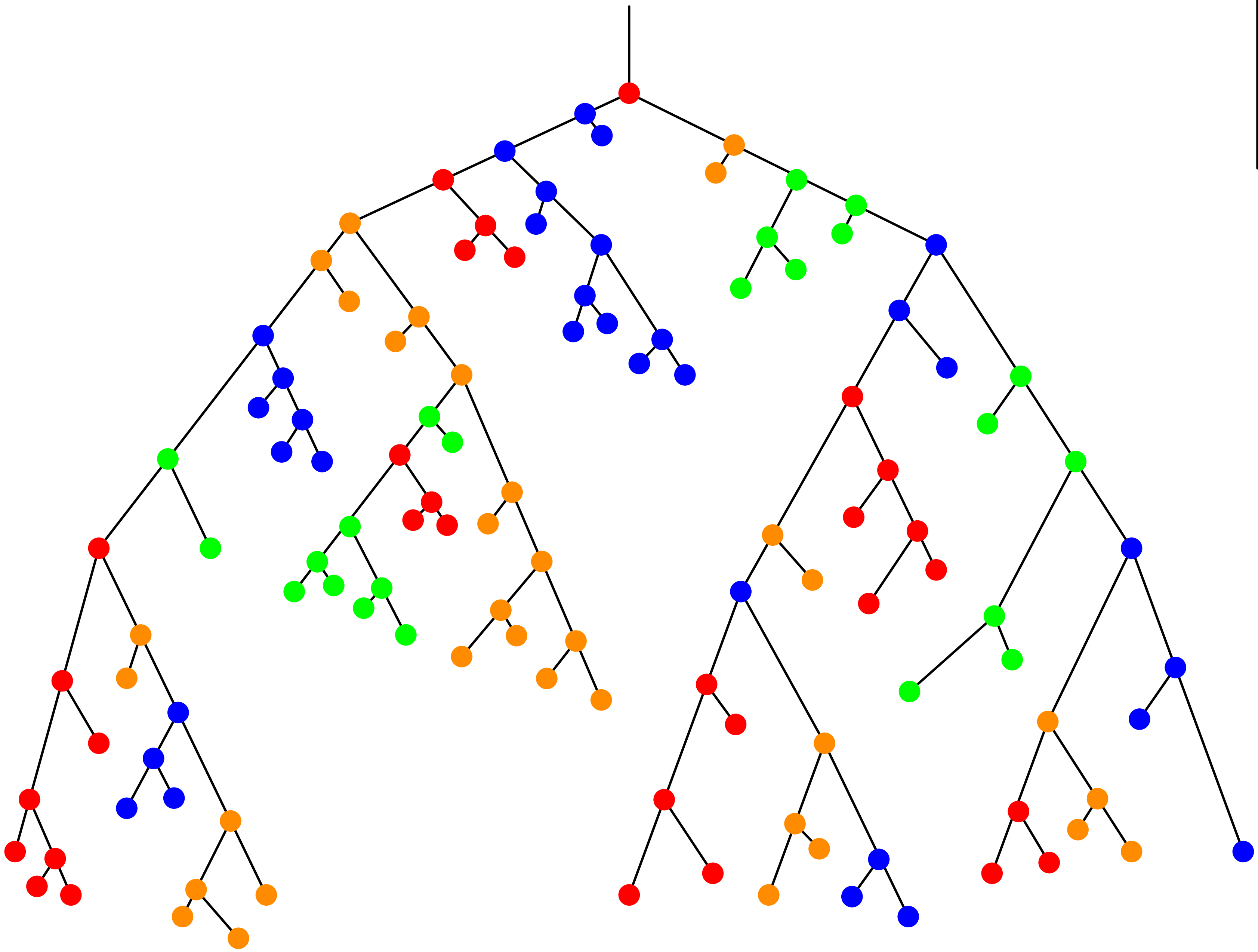
- Many ways to do this, e.g., could store for every local vertex its parent in the global tree. (Terrible for analysis.)
- **Focus on analysis**: distribute the tree to minimize communication when post-processing.

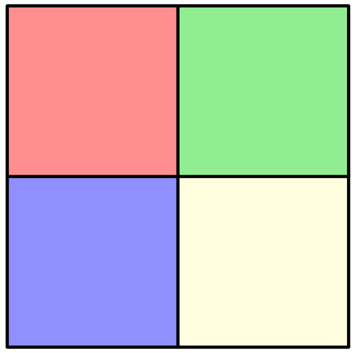
Each processor records how its local vertices fit into the global tree.

(Each branch is a connected component, so we record for every local vertex what global components it belongs to for all function values.)



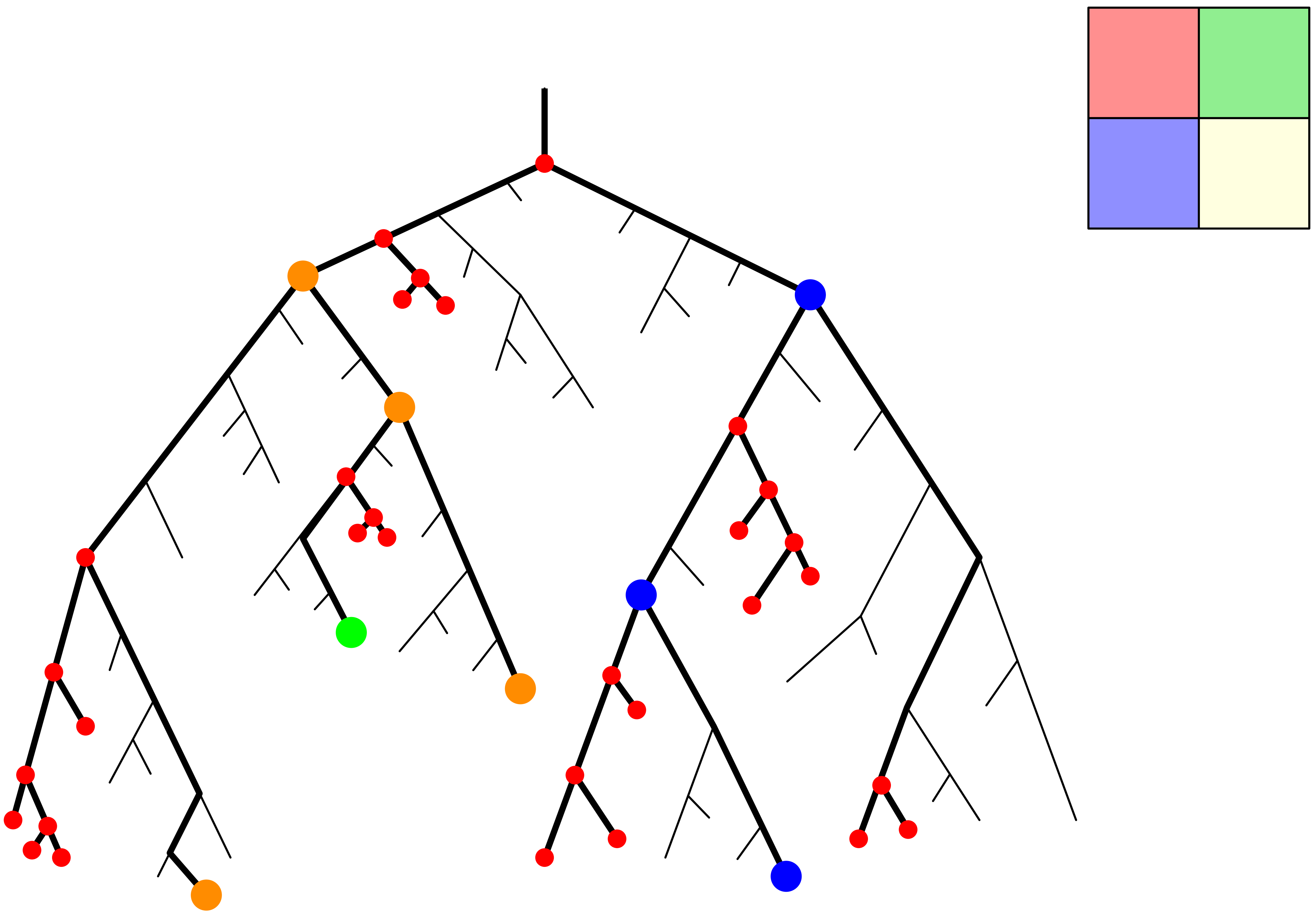
Local–Global Representation





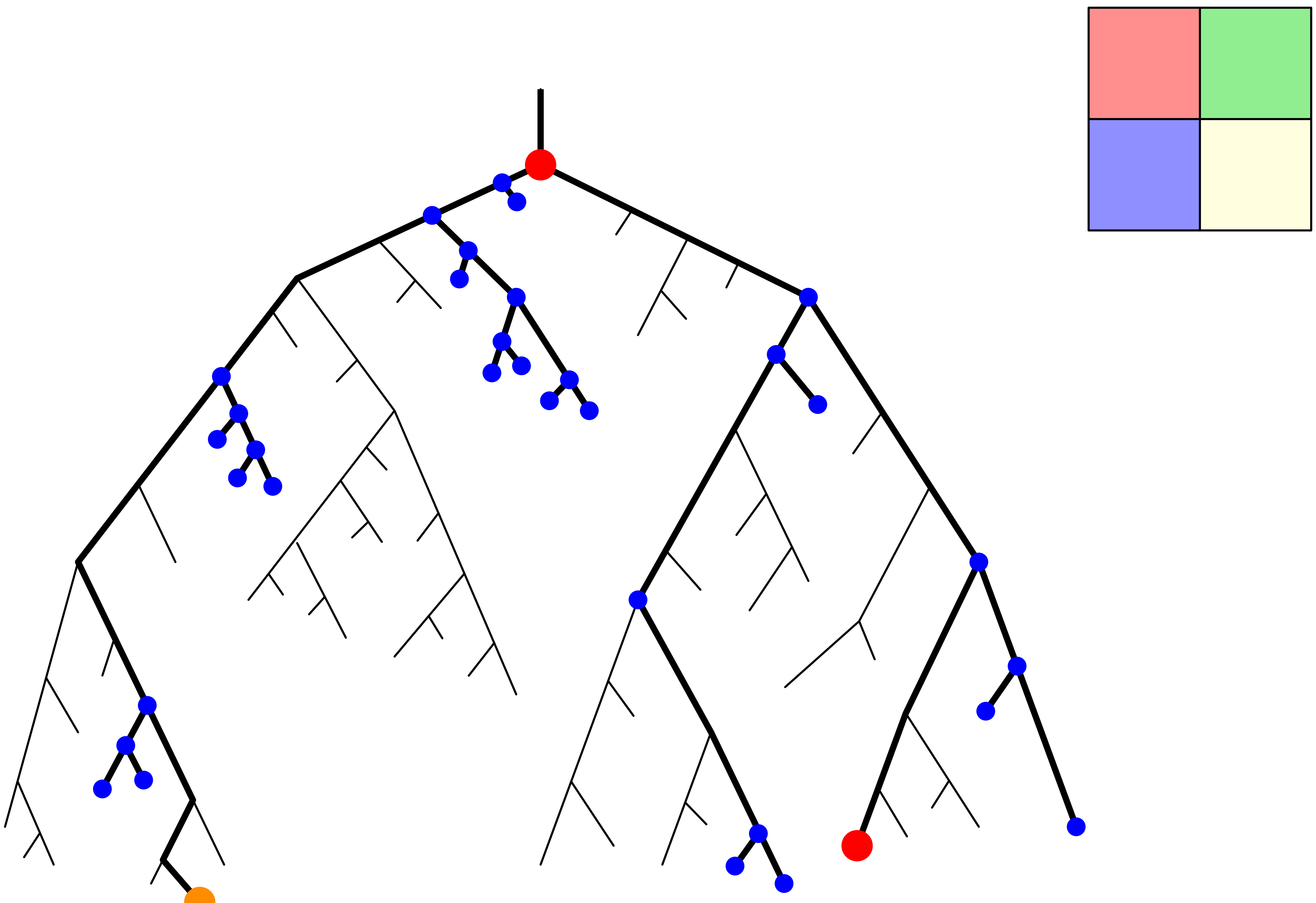
Vertex colors represent domain regions.

Local–Global Representation



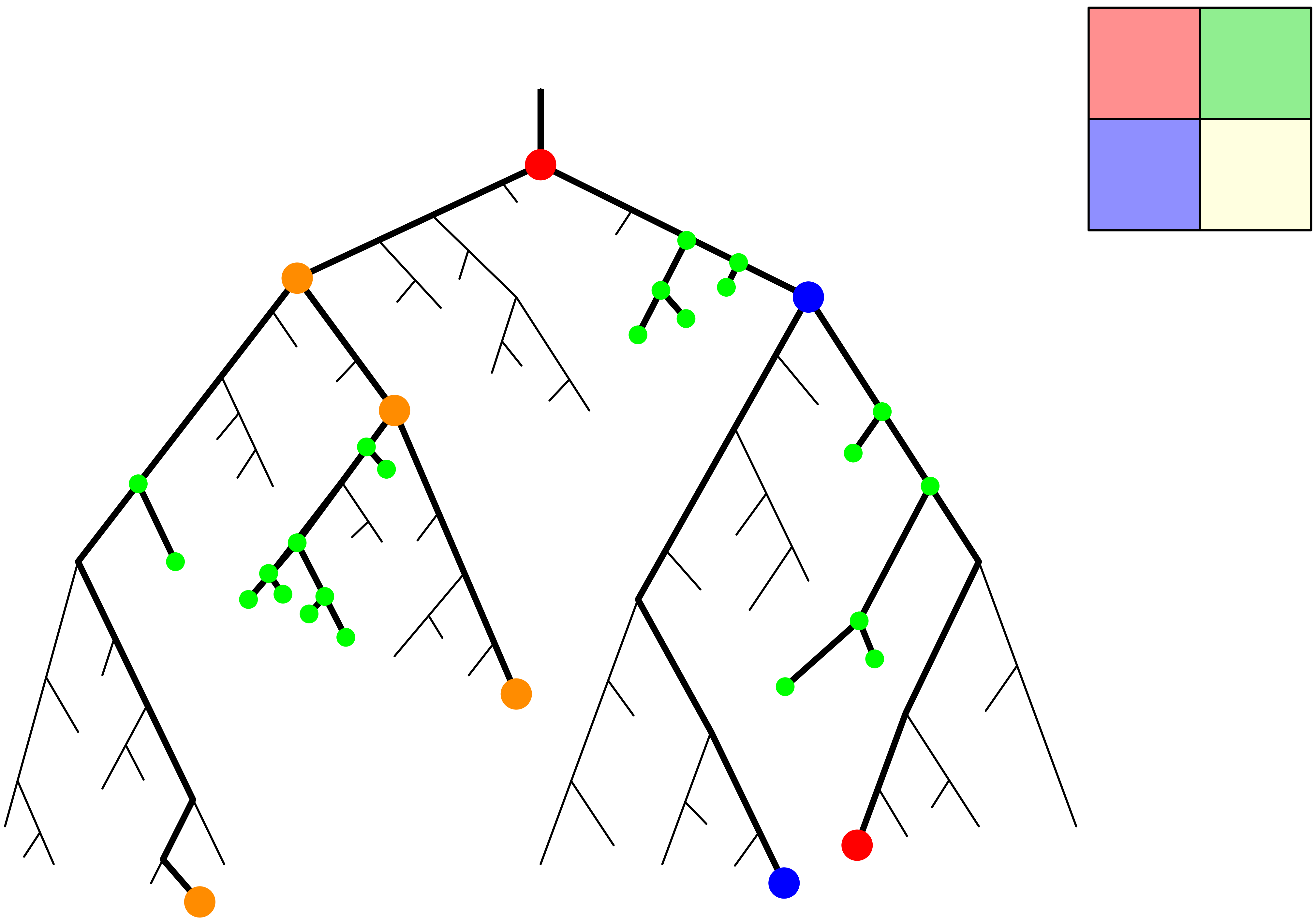
Vertex colors represent domain regions.

Local–Global Representation



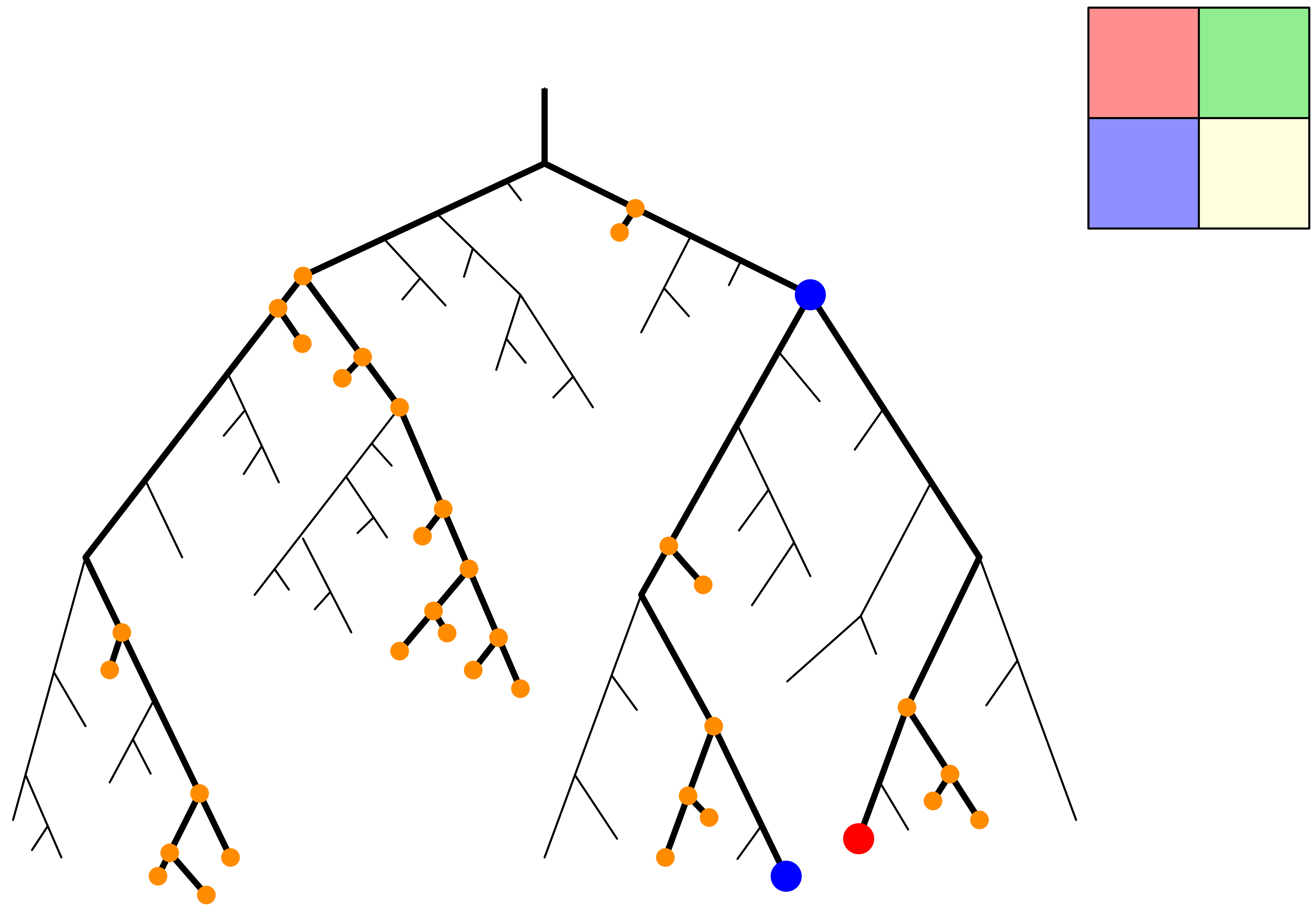
Vertex colors represent domain regions.

Local–Global Representation



Vertex colors represent domain regions.

Local–Global Representation



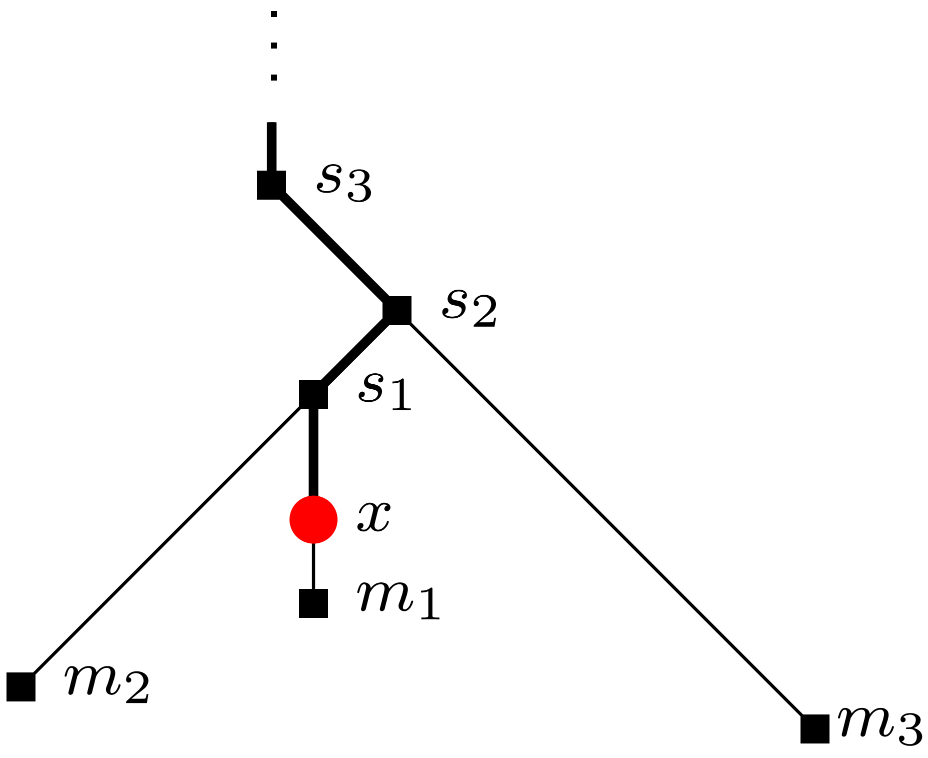
Vertex colors represent domain regions.

Analysis

Example query: compute the volumes of the sublevel set components that contain point x.

On the processor responsible for $U \ni x$:

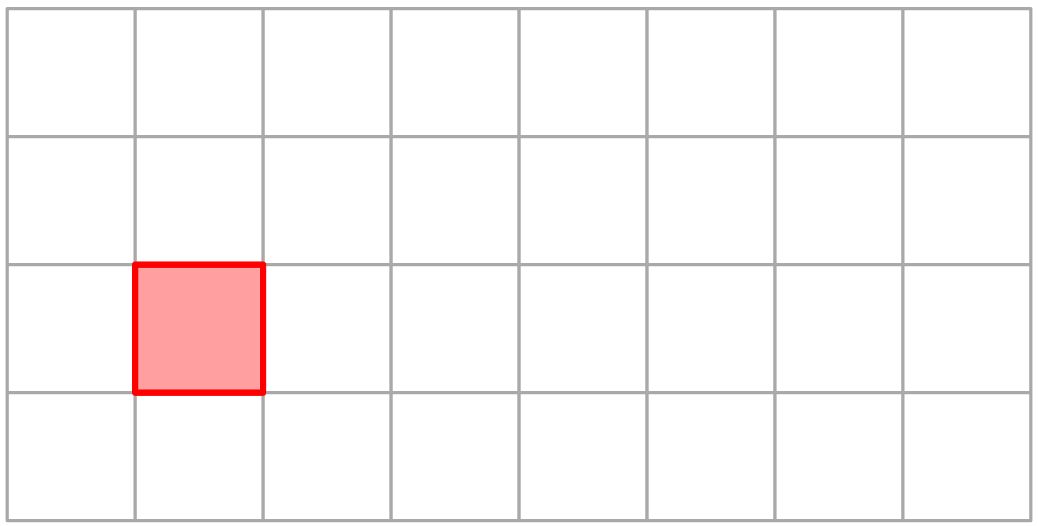
- Identify the sequence of minima and saddles $m_1, s_1, m_2, s_2, m_3, s_3, \ldots$
- broadcast this sequence to the rest of the processors
- each processor can **independently** identify its contribution to each one of these sublevel set components



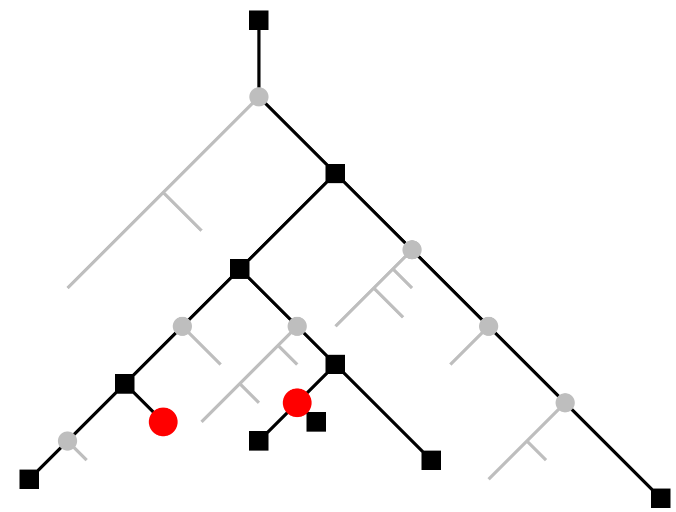
Sparse Exchange

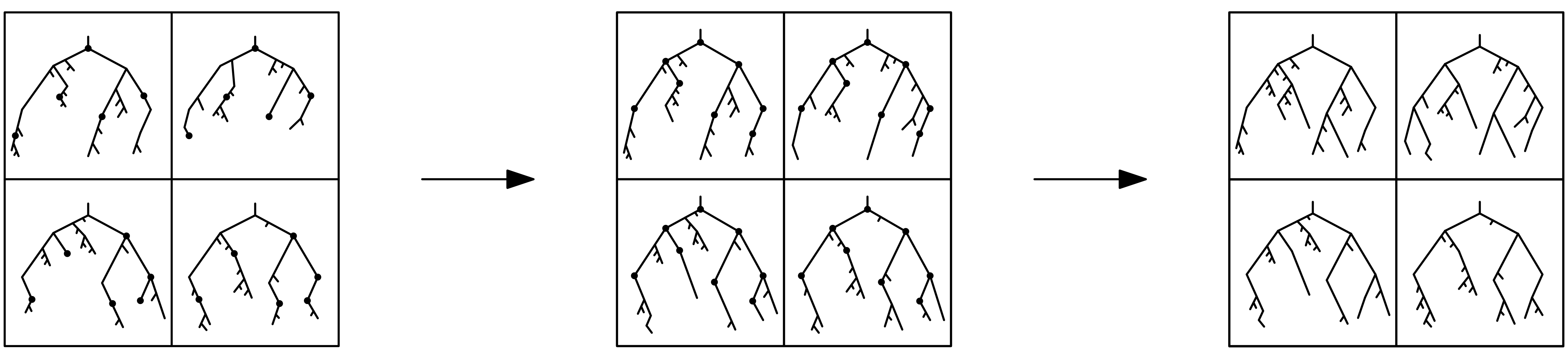
sparsification and merging can be interleaved

Each processor maintains the tree sparsified with respect to its local domain, and the boundary of its current global domain.



Once a subtree consists only of interior nodes, and its not reachable from local or boundary vertices, we can remove it.

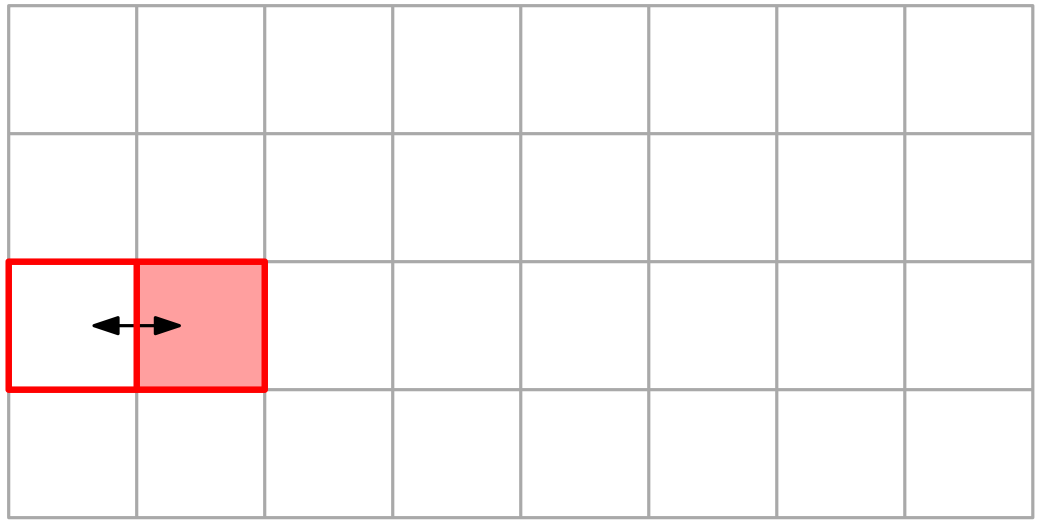




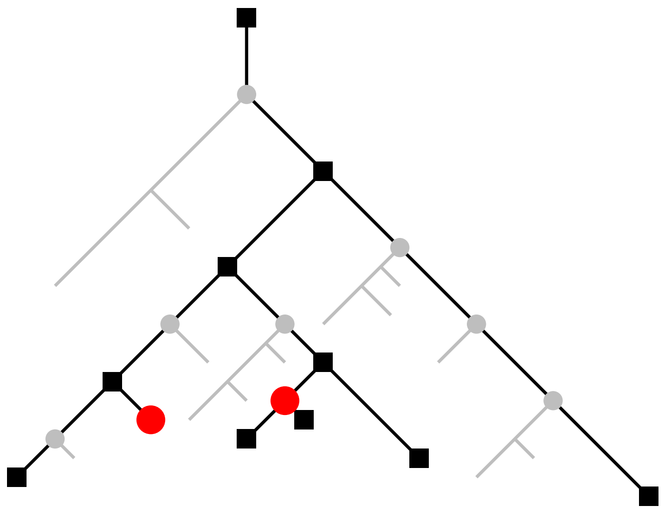
Sparse Exchange

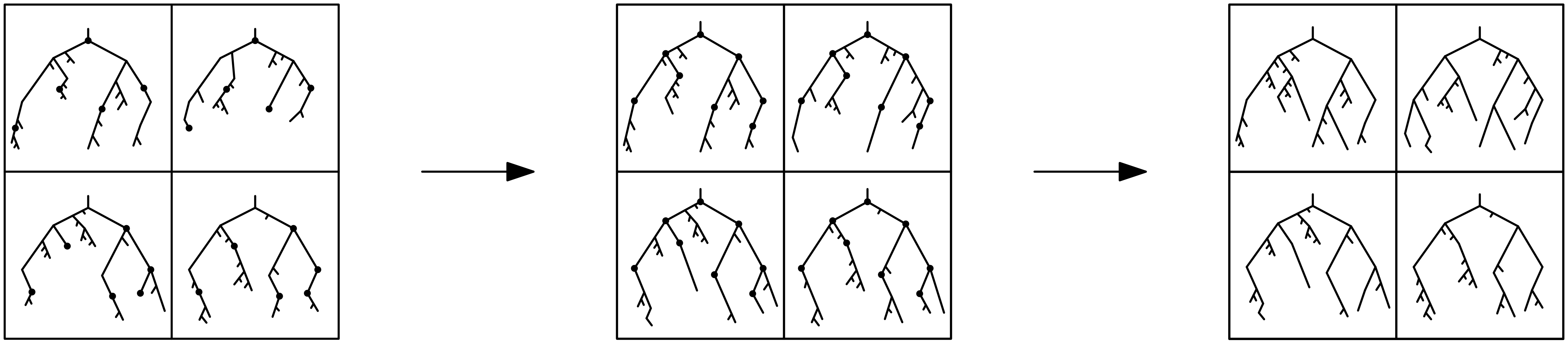
sparsification and merging can be interleaved

Each processor maintains the tree sparsified with respect to its local domain, and the boundary of its current global domain.



Once a subtree consists only of interior nodes, and its not reachable from local or boundary vertices, we can remove it.



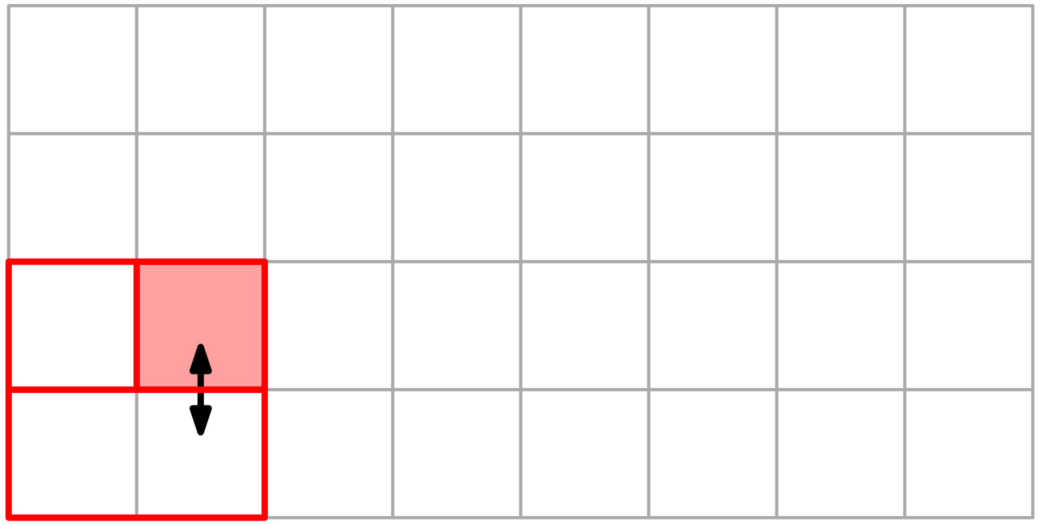


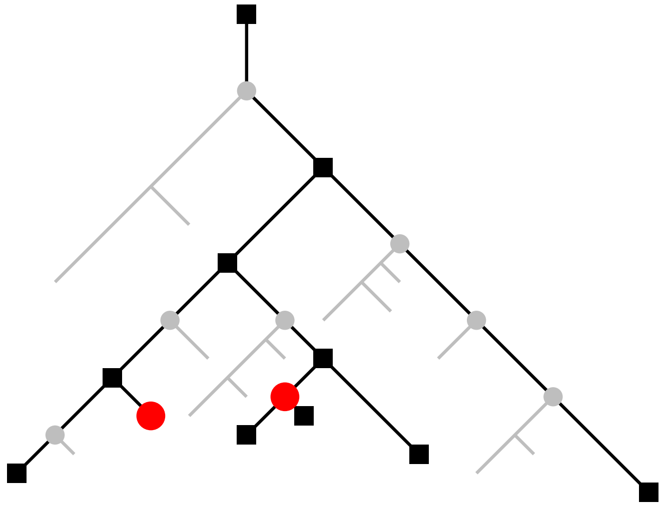
Sparse Exchange

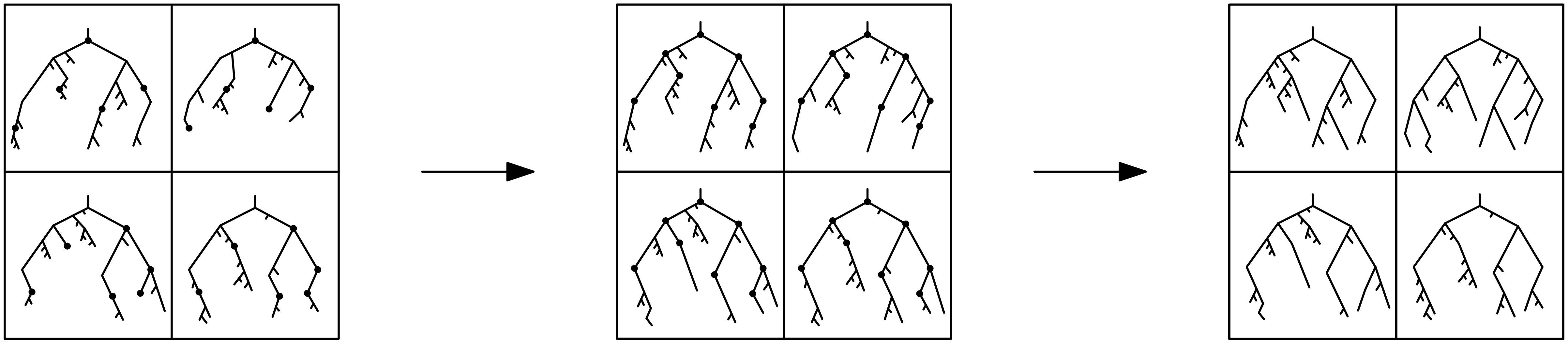
sparsification and merging can be interleaved

Each processor maintains the tree sparsified with respect to its local domain, and the boundary of its current global domain.

Once a subtree consists only of interior nodes, and its not reachable from local or boundary vertices, we can remove it.





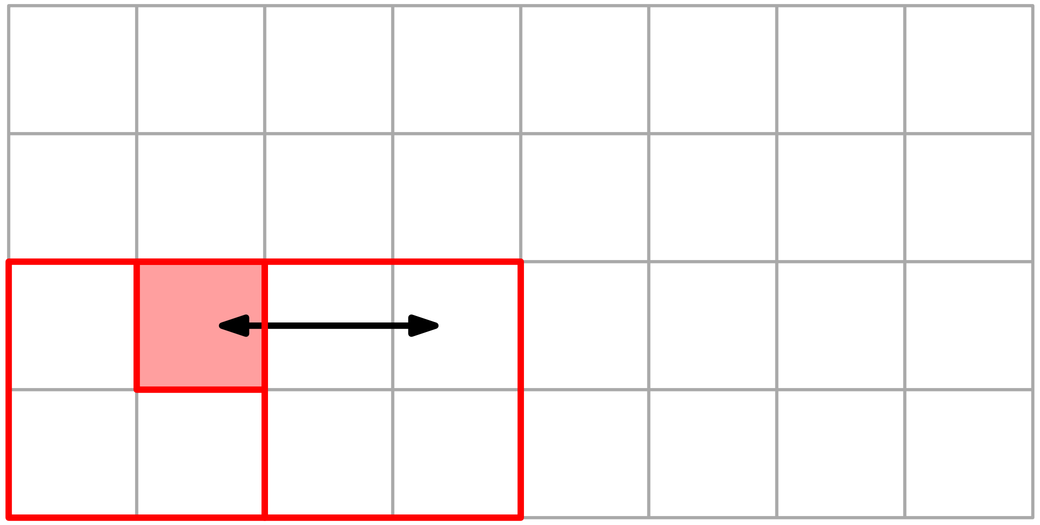


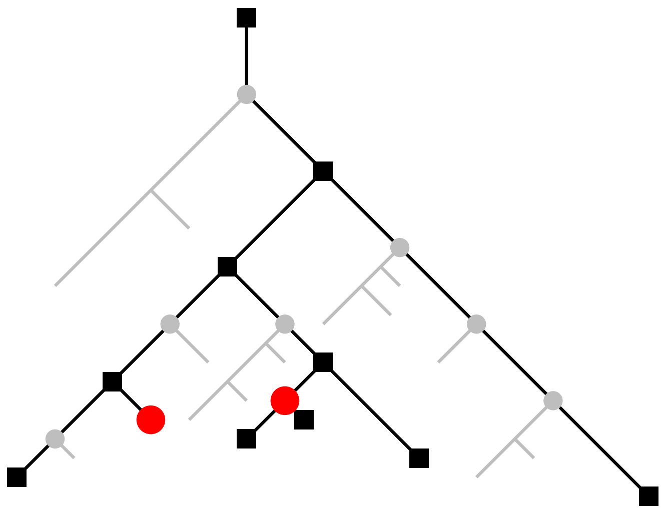
Sparse Exchange

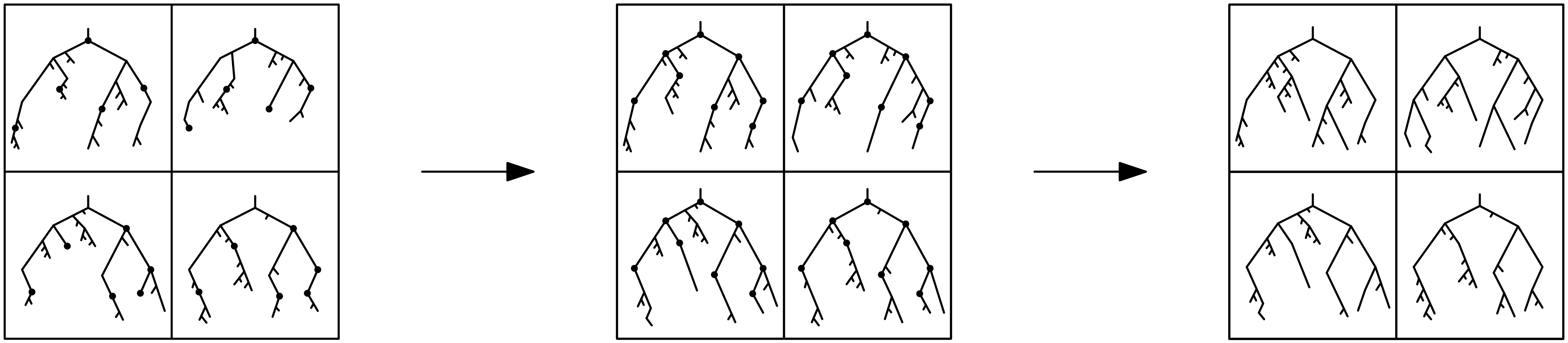
sparsification and merging can be interleaved

Each processor maintains the tree sparsified with respect to its local domain, and the boundary of its current global domain.

Once a subtree consists only of interior nodes, and its not reachable from local or boundary vertices, we can remove it.





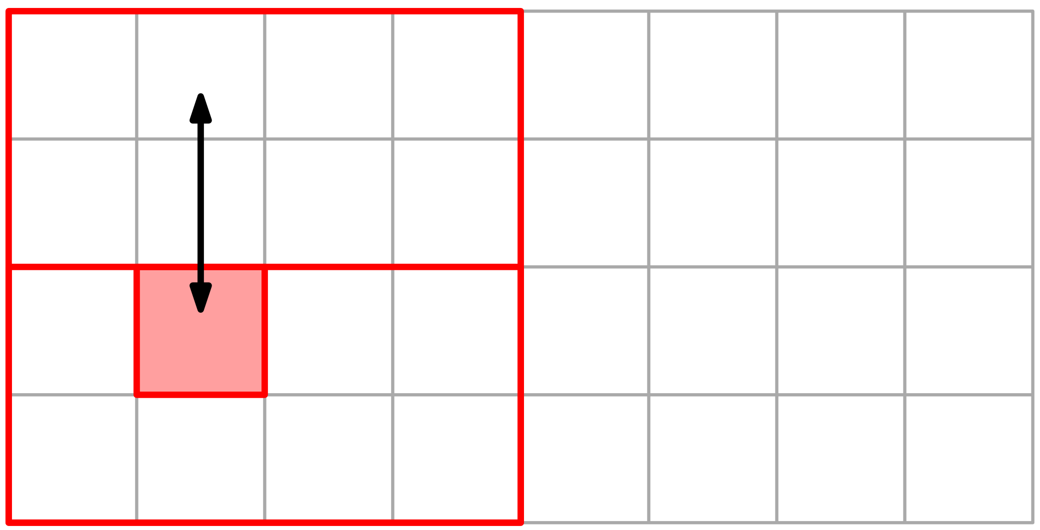


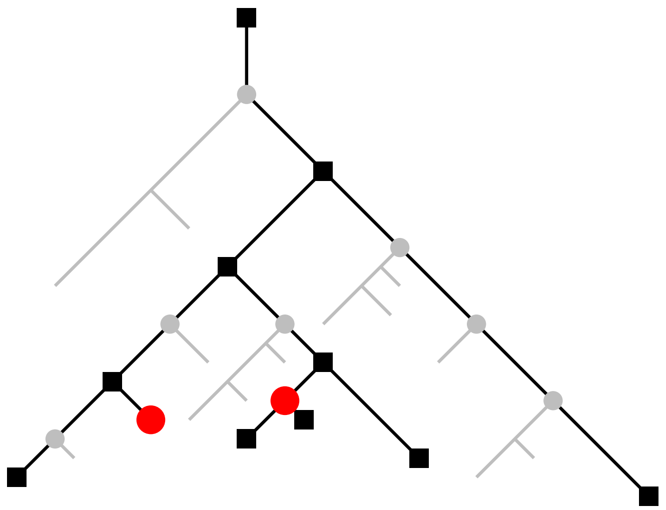
Sparse Exchange

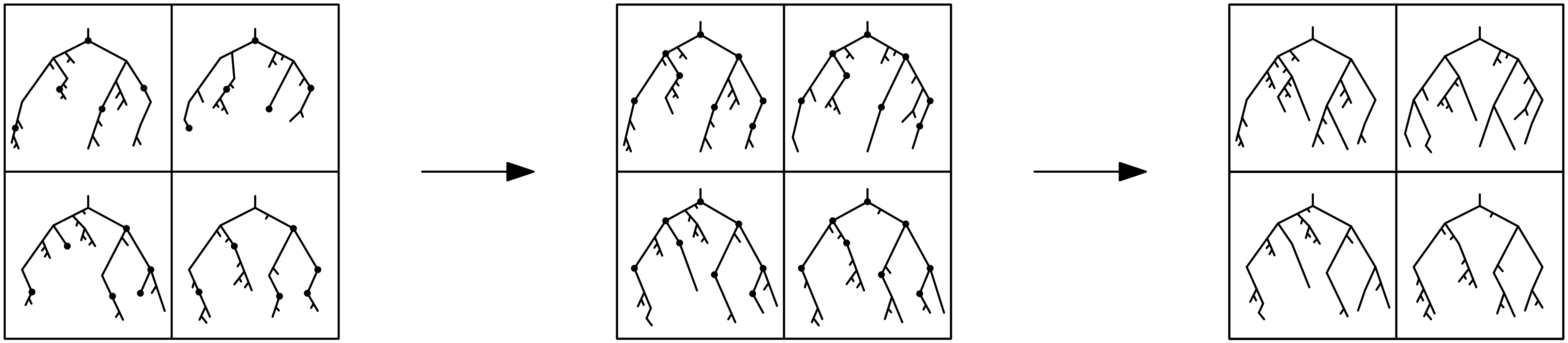
sparsification and merging can be interleaved

Each processor maintains the tree sparsified with respect to its local domain, and the boundary of its current global domain.

Once a subtree consists only of interior nodes, and its not reachable from local or boundary vertices, we can remove it.





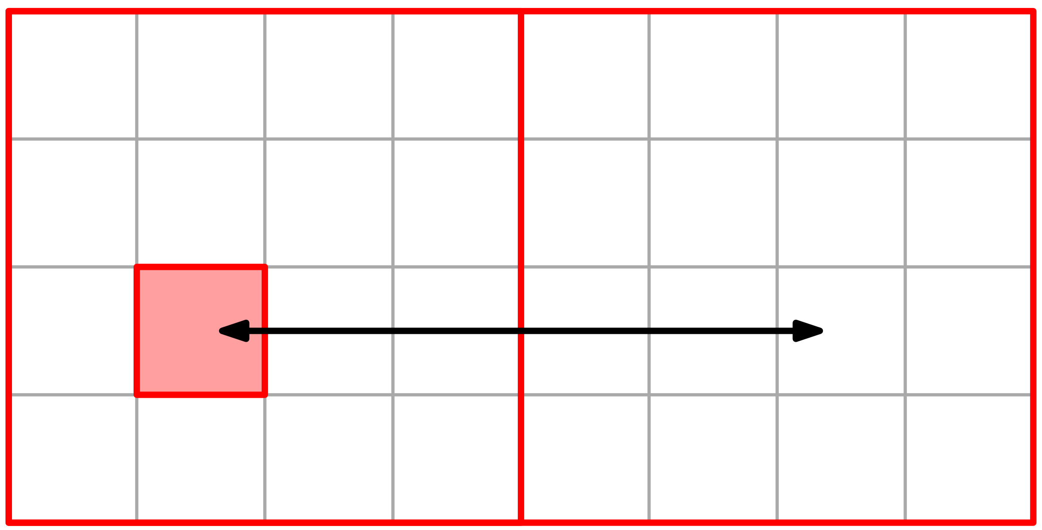


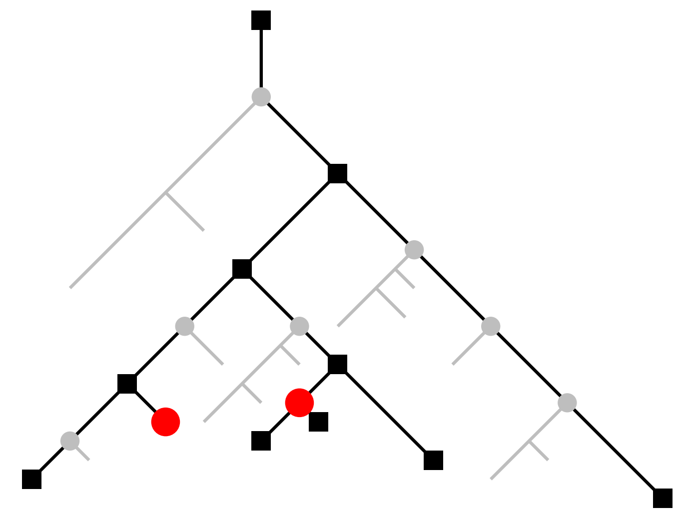
Sparse Exchange

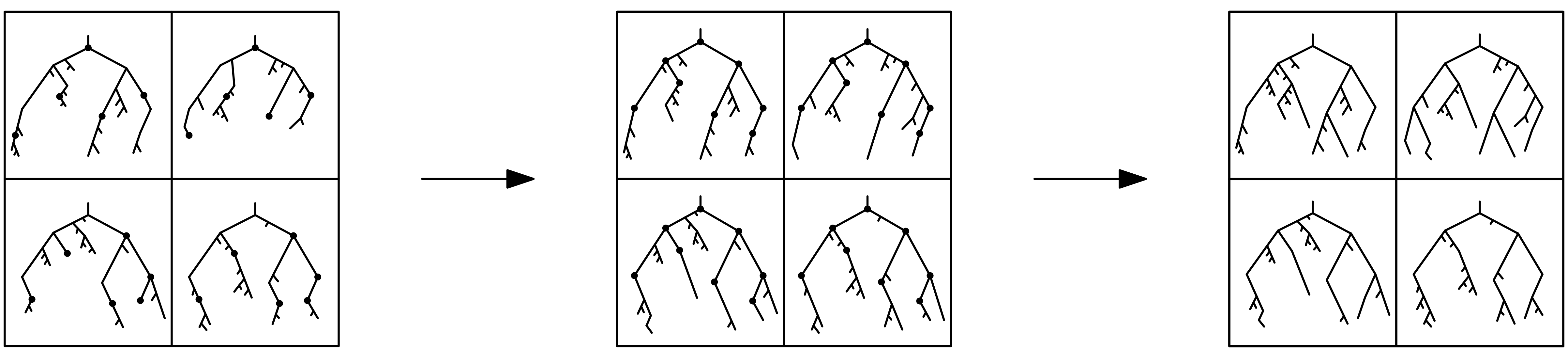
sparsification and merging can be interleaved

Each processor maintains the tree sparsified with respect to its local domain, and the boundary of its current global domain.

Once a subtree consists only of interior nodes, and its not reachable from local or boundary vertices, we can remove it.



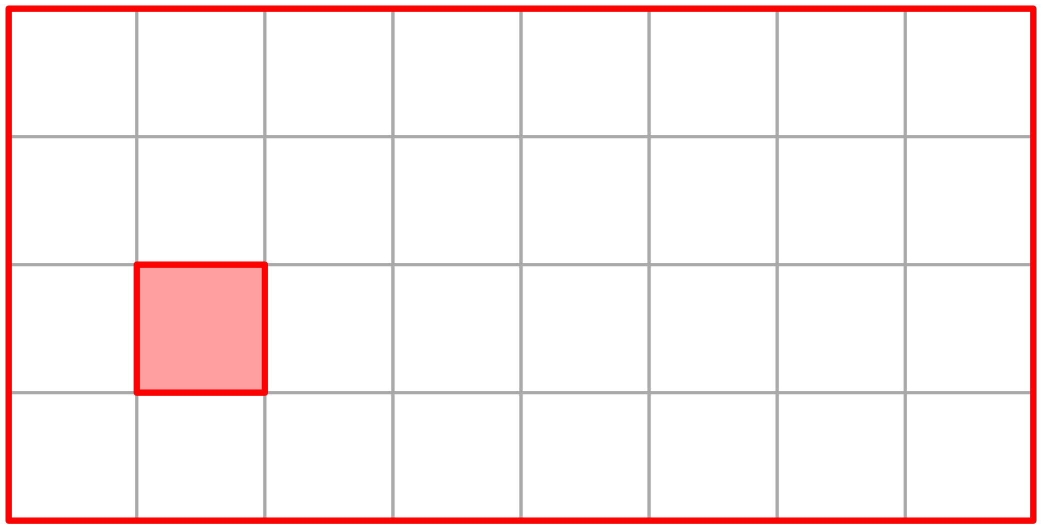




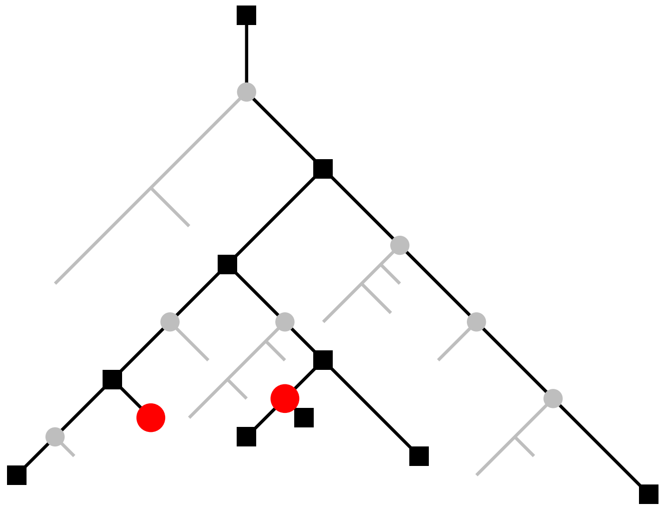
Sparse Exchange

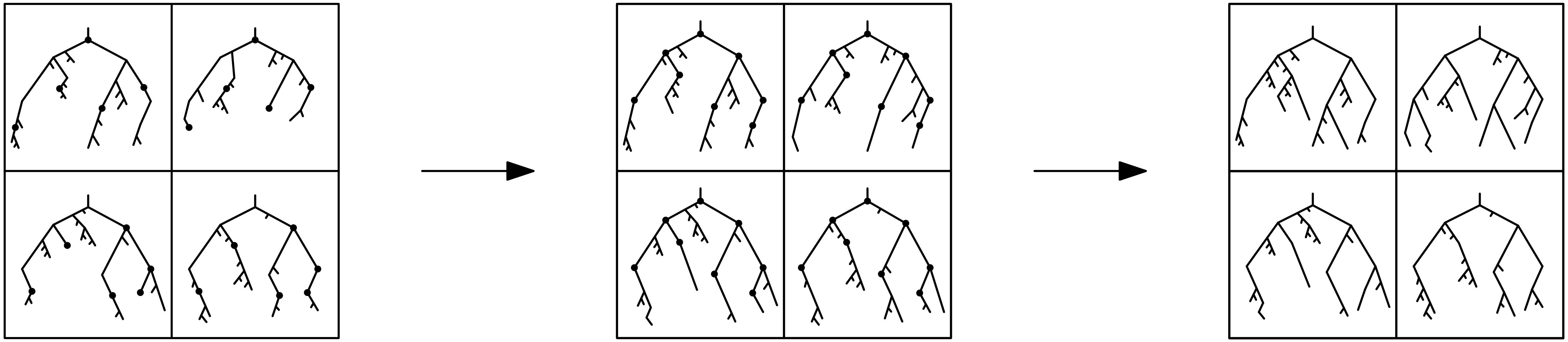
sparsification and merging can be interleaved

Each processor maintains the tree sparsified with respect to its local domain, and the boundary of its current global domain.



Once a subtree consists only of interior nodes, and its not reachable from local or boundary vertices, we can remove it.

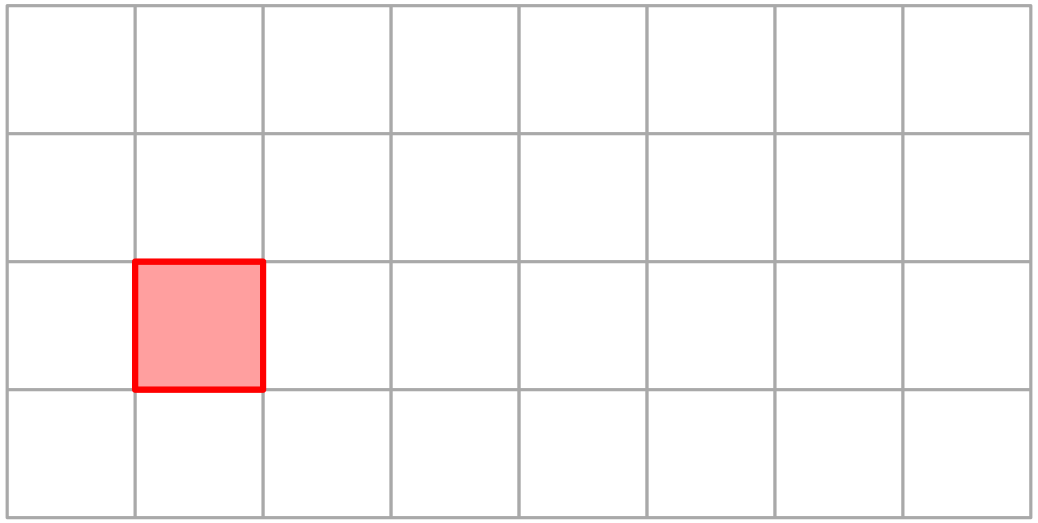




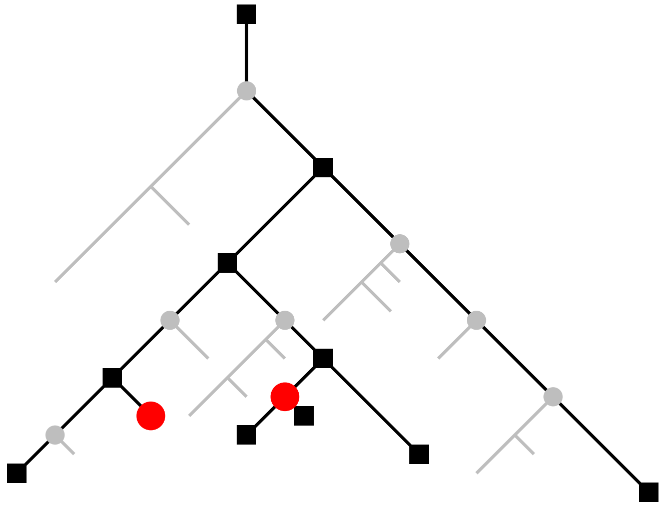
Sparse Exchange

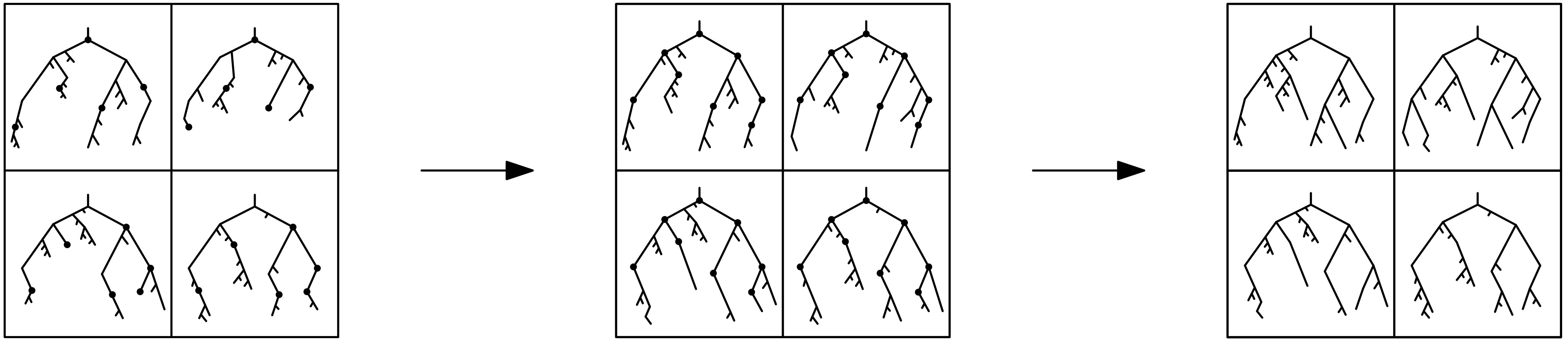
sparsification and merging can be interleaved

Each processor maintains the tree sparsified with respect to its local domain, and the boundary of its current global domain.

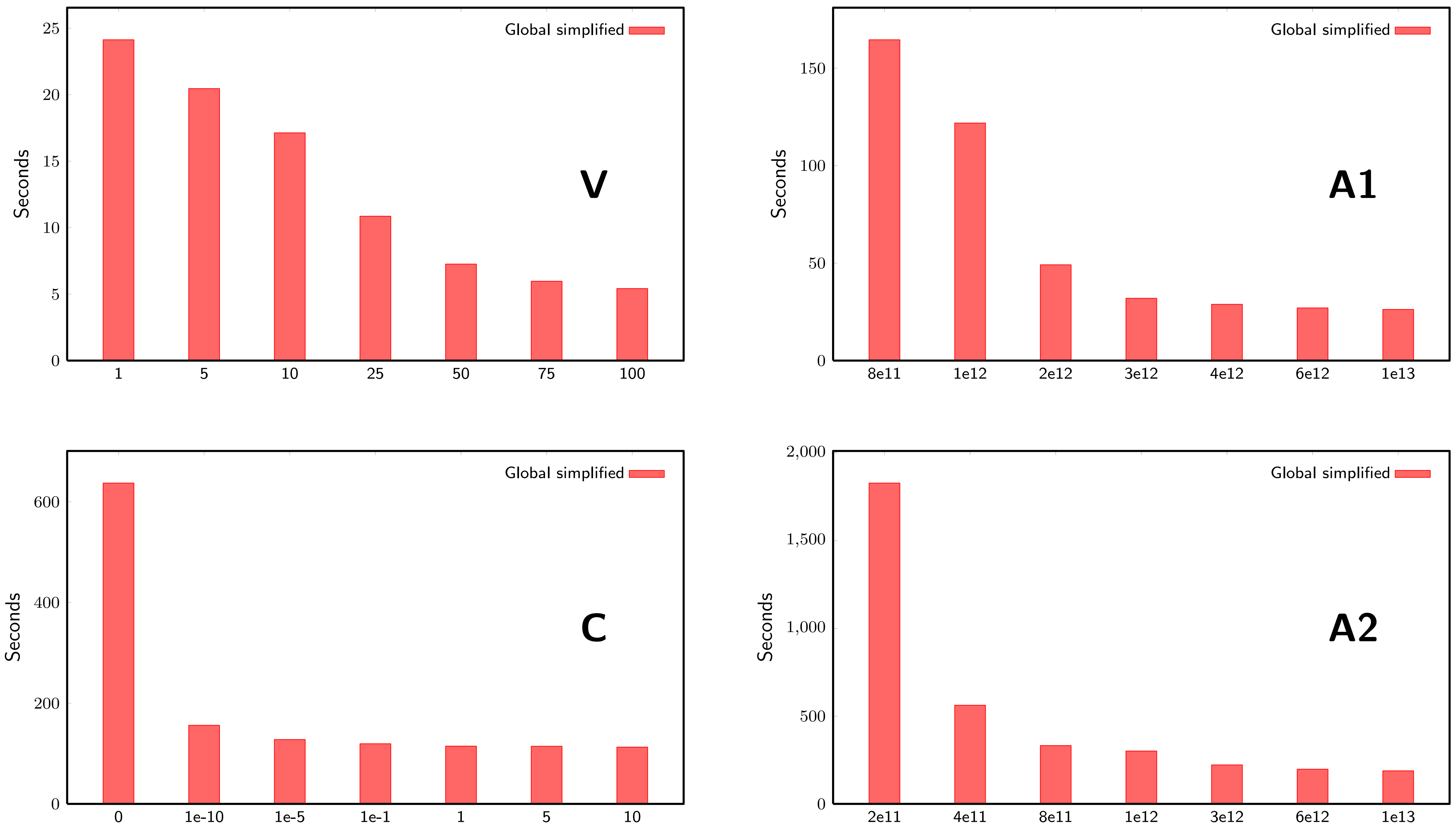


Once a subtree consists only of interior nodes, and its not reachable from local or boundary vertices, we can remove it.



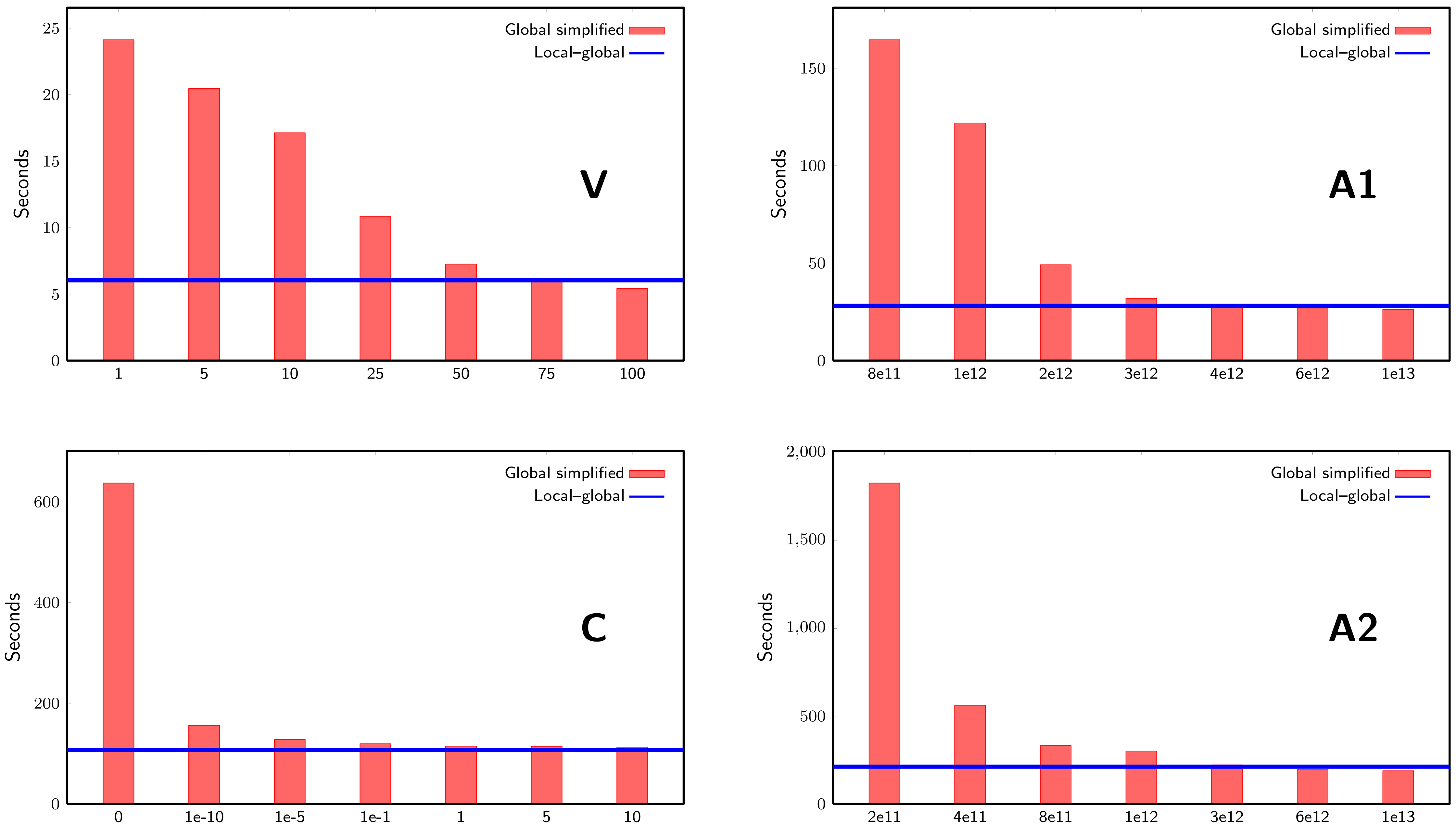


Timings (using 512 processors)



A2	(2048^3):	astrophysics simulation
C	$(1024^2 \times 2048)$:	combustion simulation
A1	(1024^3):	astrophysics simulation
V	(512^3):	rotational angiography scan

Timings (using 512 processors)



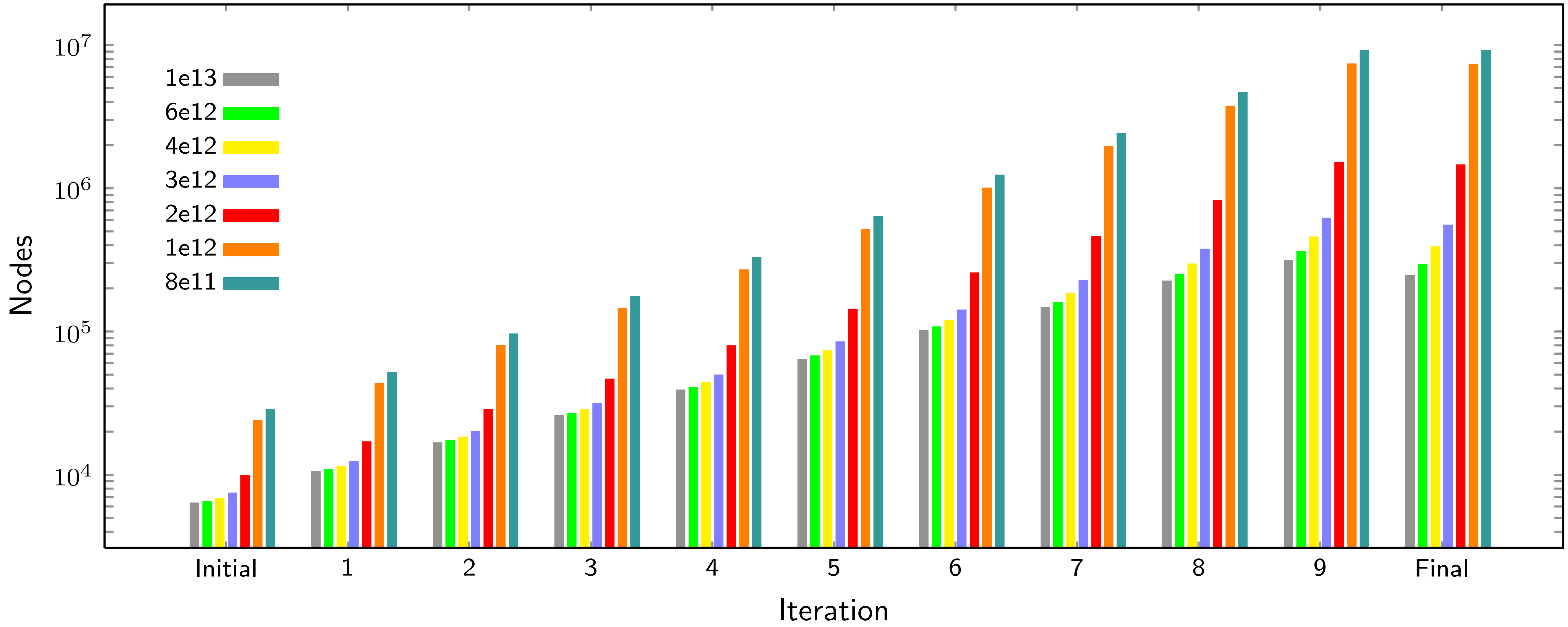
Almost as fast to compute as the most aggressive simplification, but doesn't lose information.

A2	(2048^3):	astrophysics simulation
C	$(1024^2 \times 2048)$:	combustion simulation
A1	(1024^3):	astrophysics simulation
V	(512^3):	rotational angiography scan

Tree growth (using 512 processors)

Input: $1,024^3$ grid of particle density (astrophysics data).

Largest tree size during each iteration on any processor.

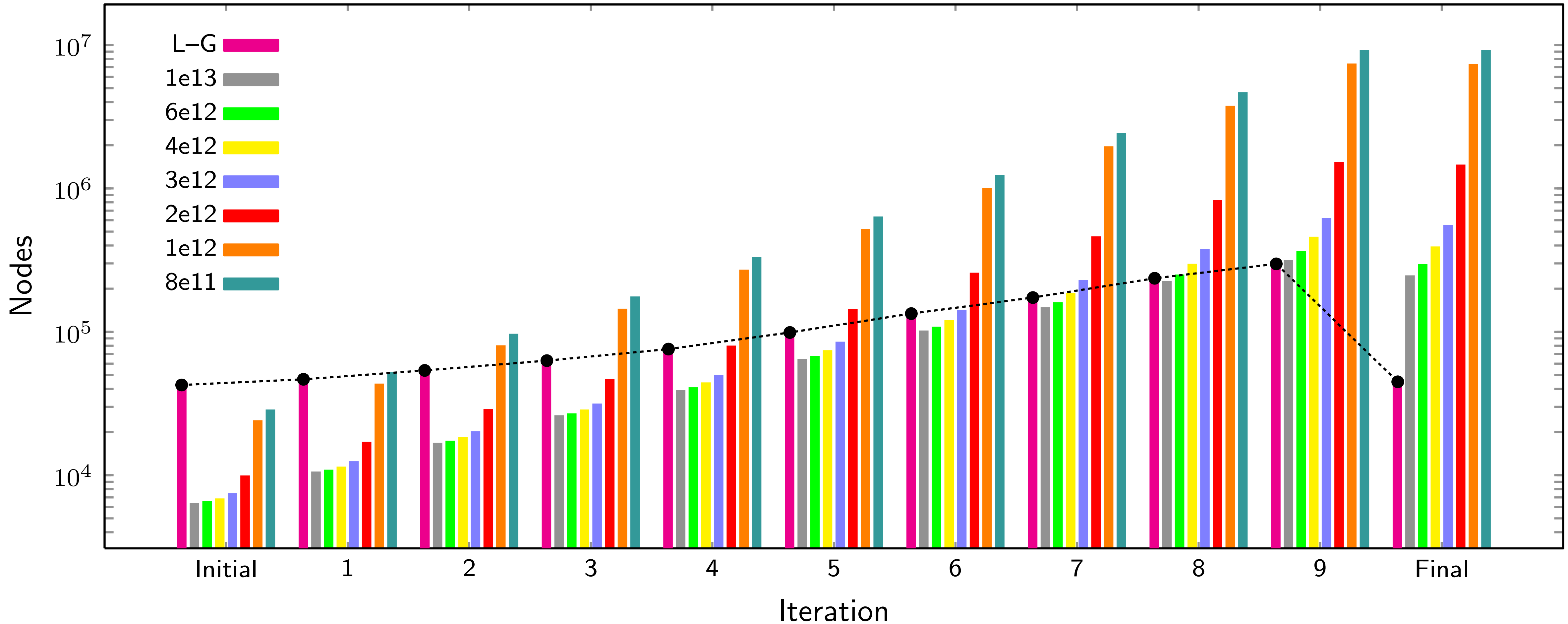


Tree growth

(using 512 processors)

Input: $1,024^3$ grid of particle density (astrophysics data).

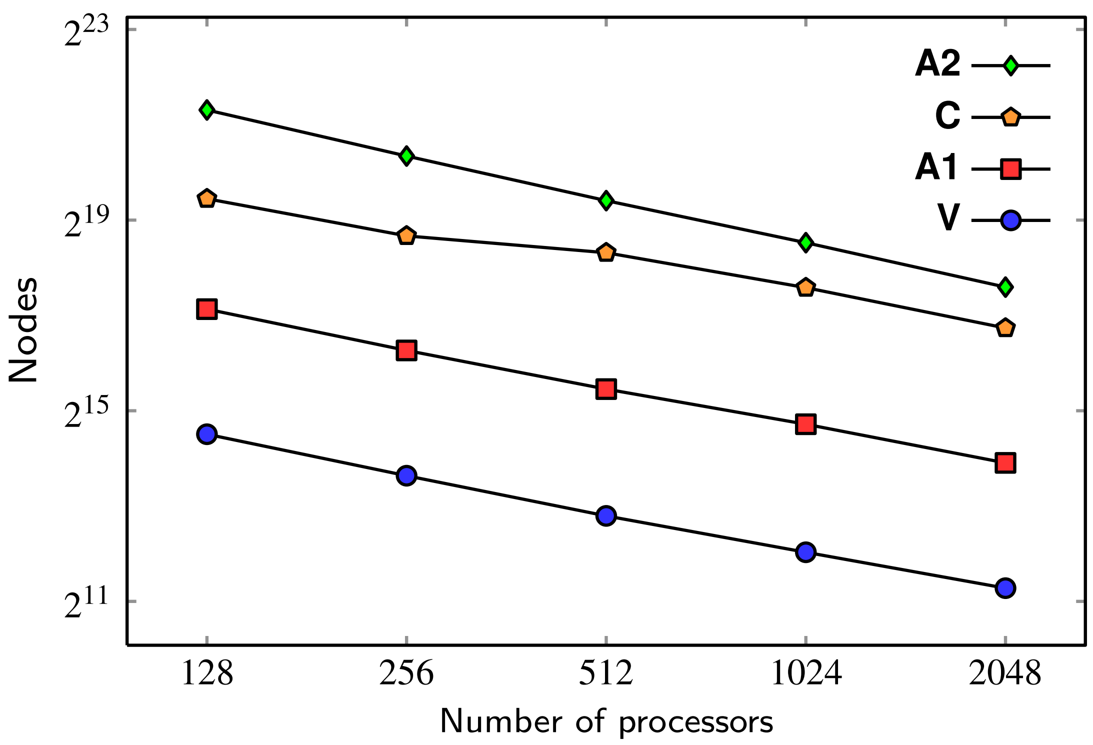
Largest tree size during each iteration on any processor.



End result: full merge tree (no information loss), but each processor has to store only a small representation.

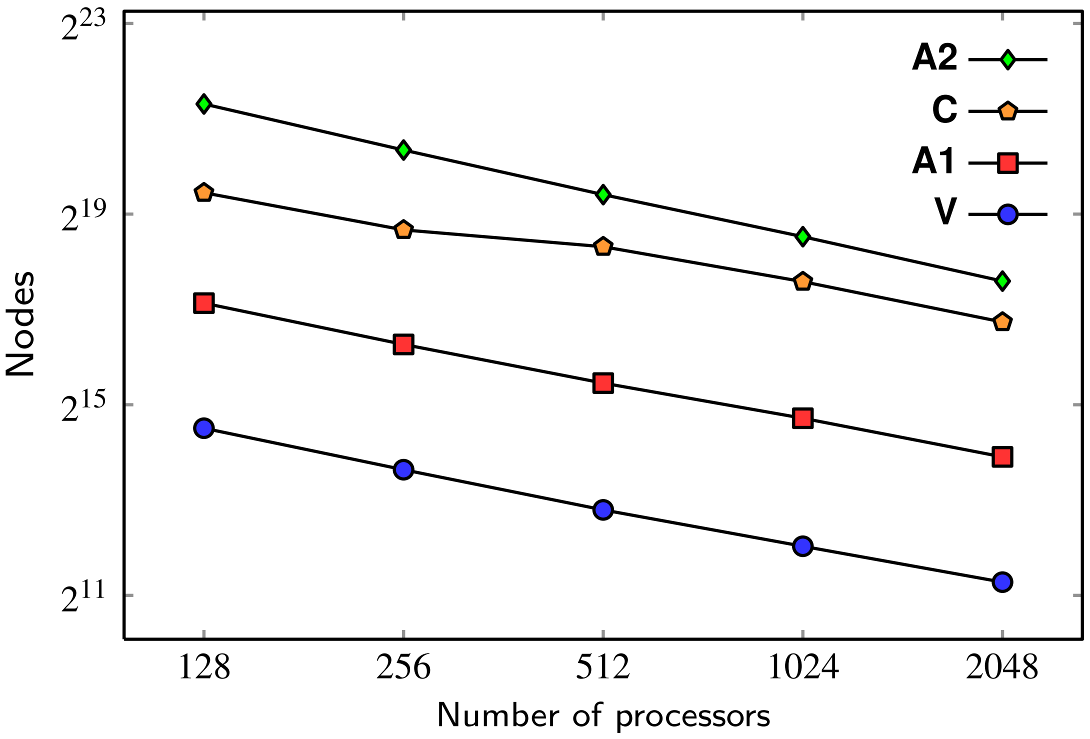
Results

Final tree sizes as we increase the number of processors (these serve as the input to the analysis routines):

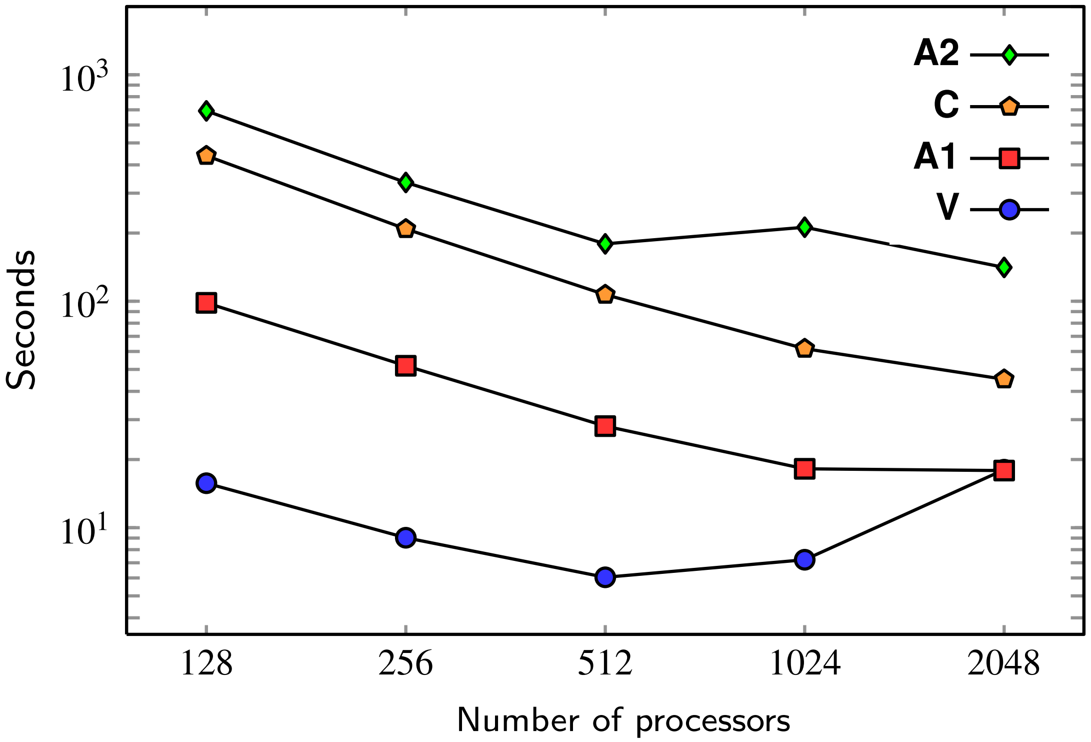


Results

Final tree sizes as we increase the number of processors (these serve as the input to the analysis routines):

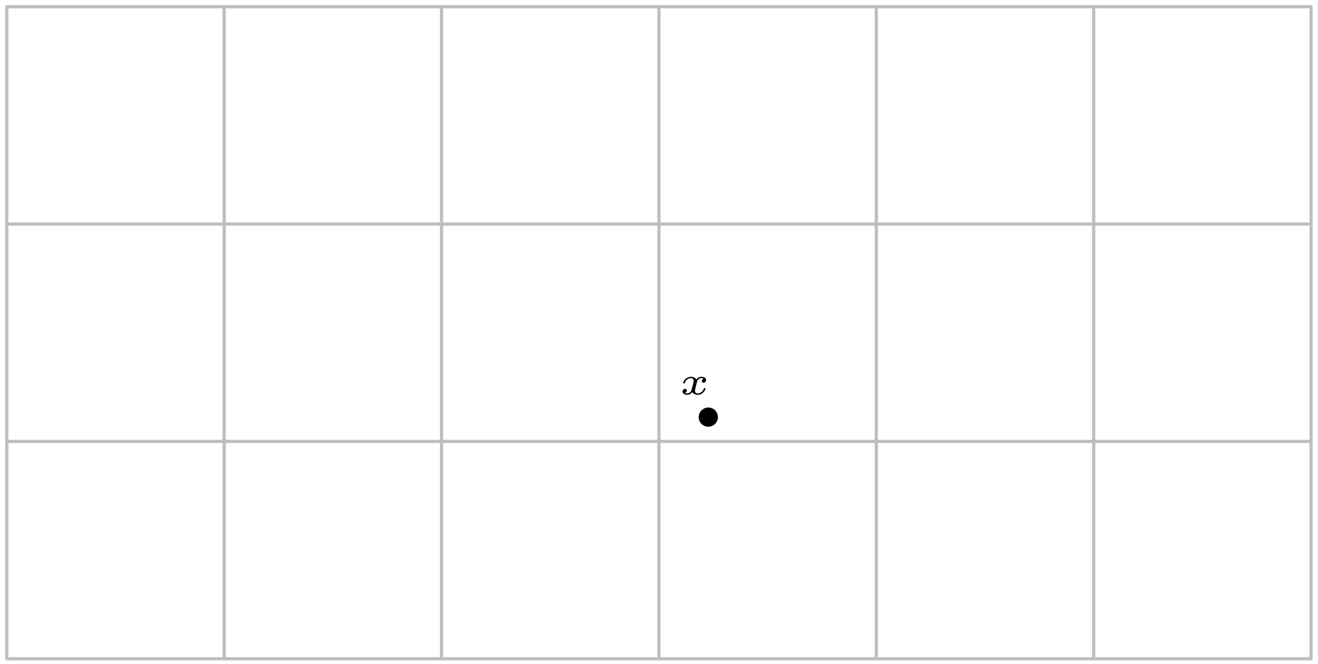


Times to compute this representation:



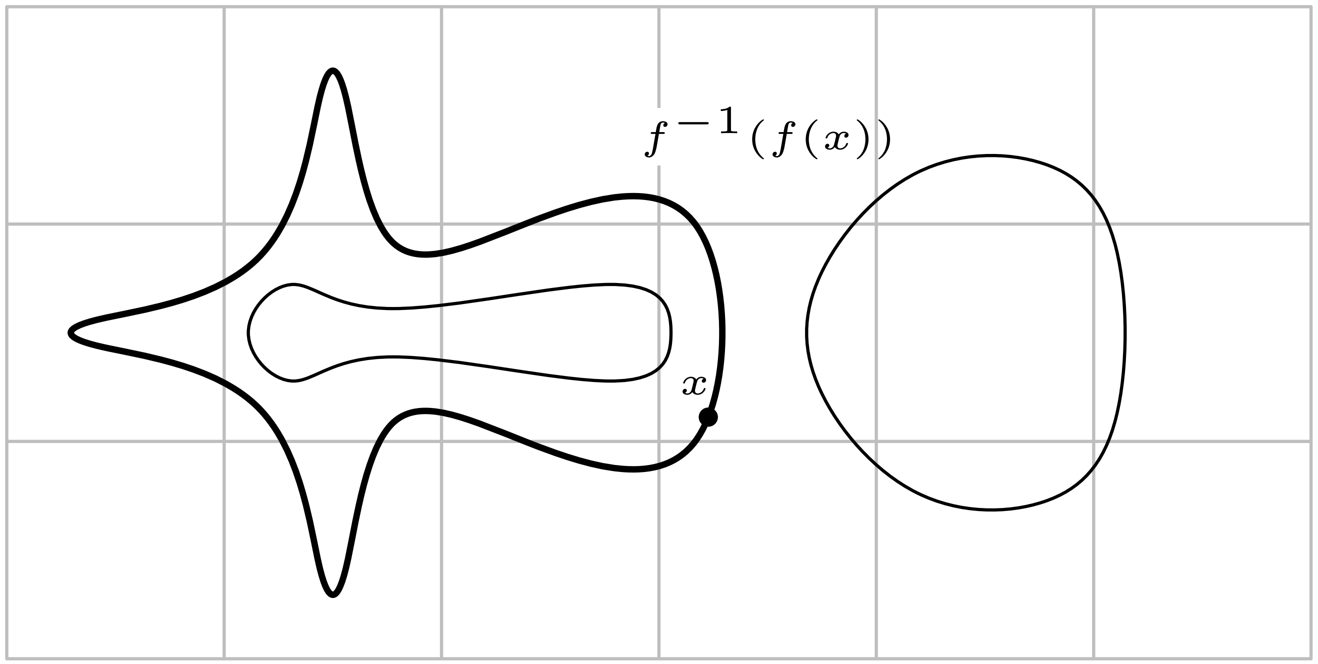
Analysis routine: levelset component extraction

Problem:
User chooses a point x, extract component of $f^{-1}(f(x))$ that contains x.



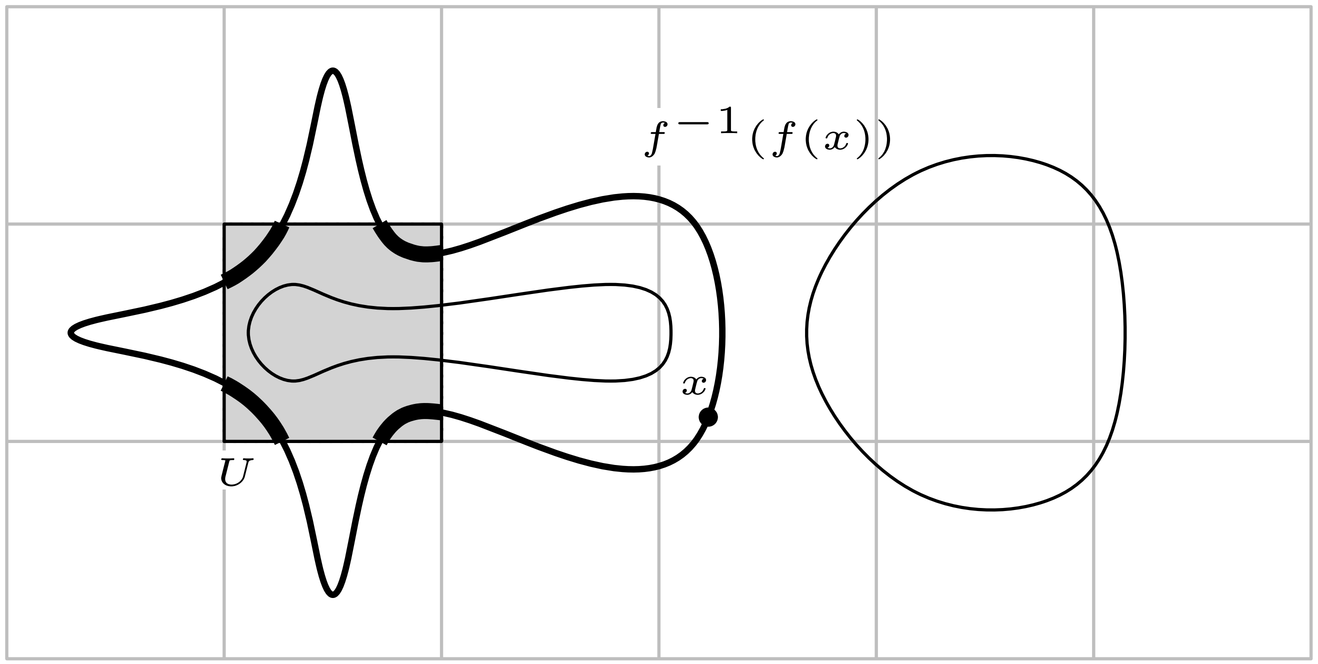
Analysis routine: levelset component extraction

Problem:
User chooses a point x, extract component of $f^{-1}(f(x))$ that contains x.



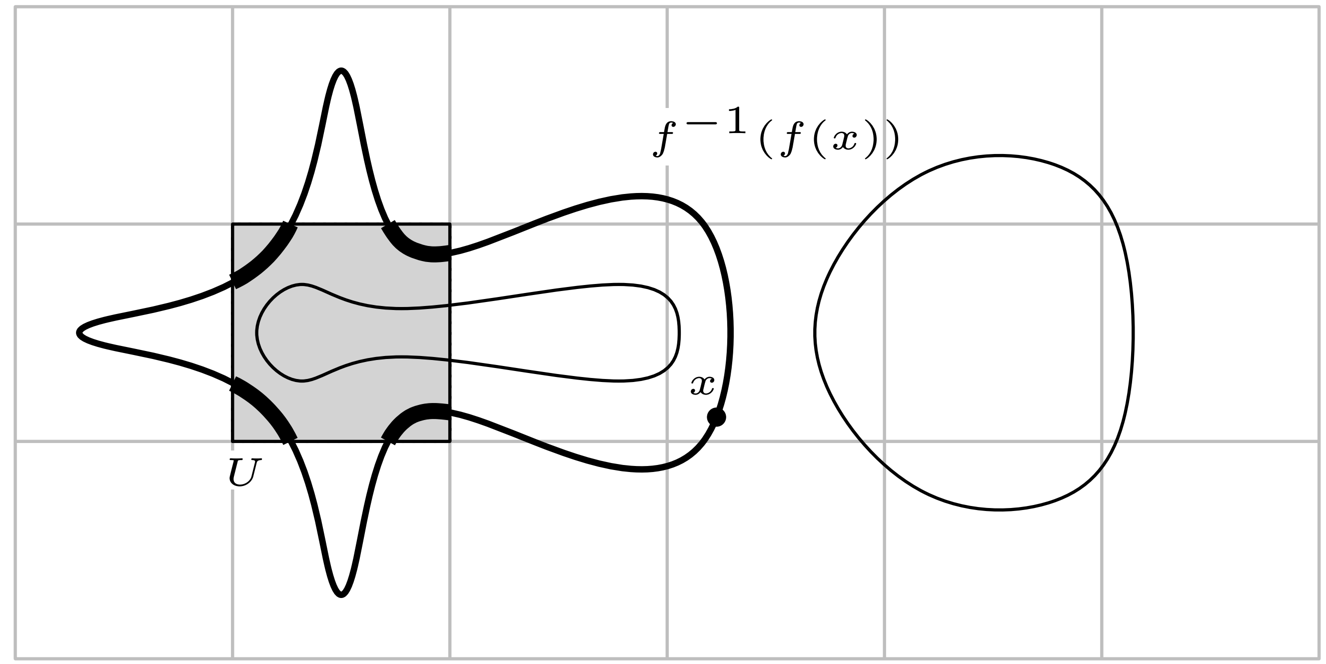
Analysis routine: levelset component extraction

Problem:
User chooses a point x, extract component of $f^{-1}(f(x))$ that contains x.



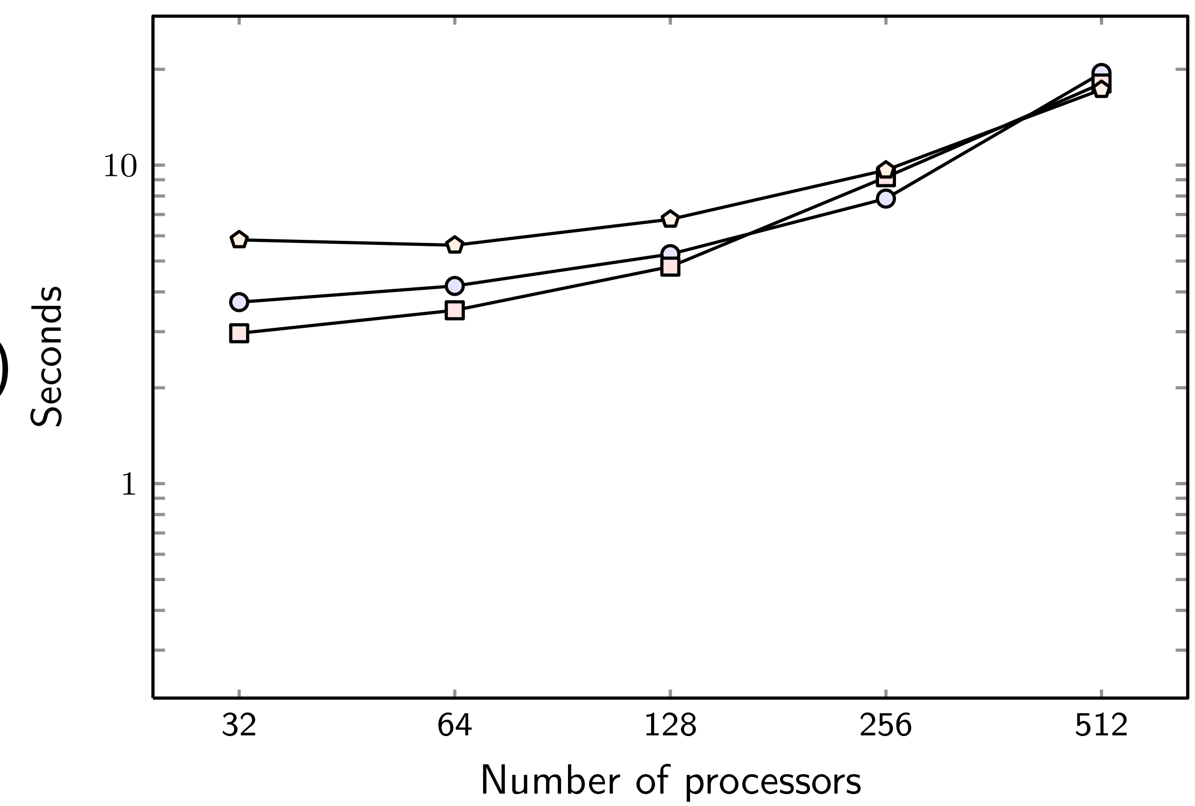
Analysis routine: levelset component extraction

Problem:
User chooses a point x,
extract component of
$f^{-1}(f(x))$ that contains x.

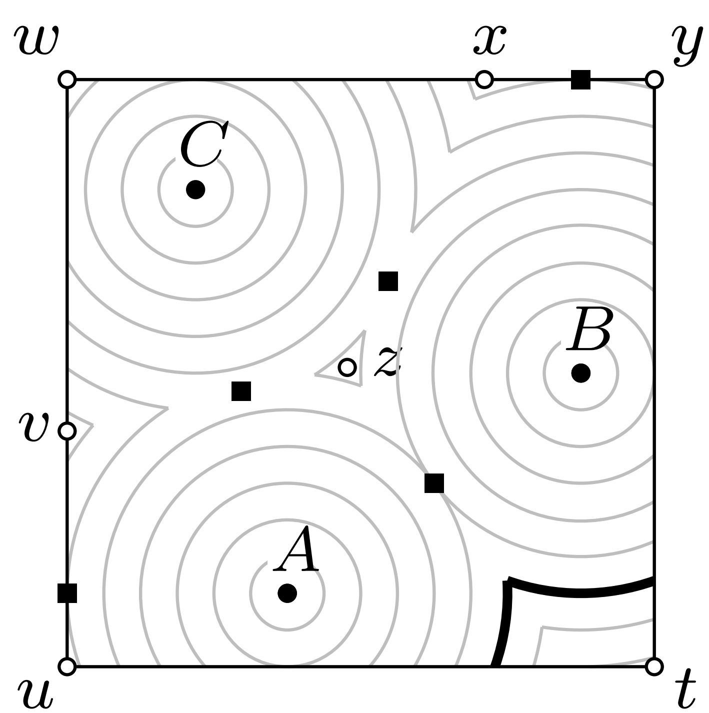


Input: 512^3 grids, medical images

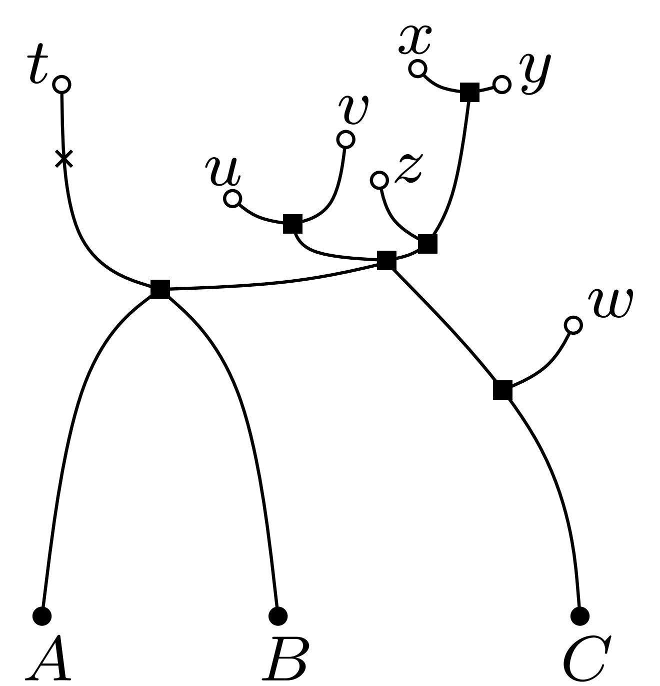
VisIt (state of the art visualization software) extracts the components and then labels them.



Contour Trees



Distance function to $\{A, B, C\}$.



Contour tree of the function.

$f : \mathbb{X} \to \mathbb{R}$

Two points are equivalent, $x \sim y$, if $f(x) = f(y)$ and they belong to the same component of the levelset $f^{-1}(f(x))$.

Reeb graph $=$ quotient space $\mathbb{X}/\sim$ $=$ continuously contract contours to points

If $\mathbb{X}$ is simply connected, Reeb graph is called a **contour tree**.

[Carr, Snoeyink, Axen '03]:

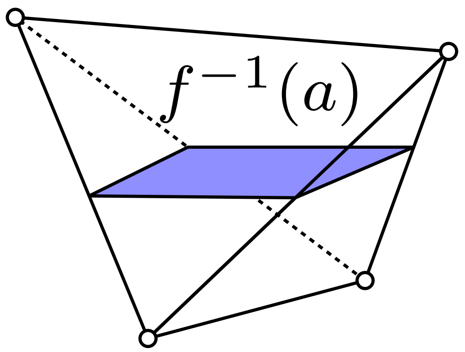
compute contour tree from merge trees of f and $-f$ in linear time.

Merge trees of f and $-f$ contain the information that we want.

Contours

Problem: Given a point x, extract component of $f^{-1}(f(x))$ that contains x.

To extract the full contour, intersect every maximal simplex with the levelset.

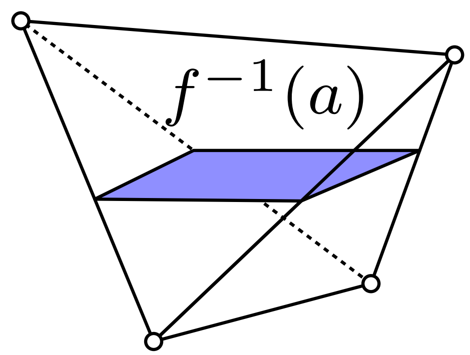


But we want to only report the component that contains x.

Contours

Problem: Given a point x, extract component of $f^{-1}(f(x))$ that contains x.

To extract the full contour, intersect every maximal simplex with the levelset.



But we want to only report the component that contains x.

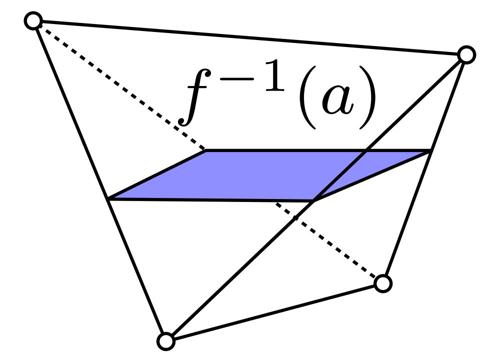
Idea: Local–global represenation determines a globally unique component ID without any communication. On simply connected domains, sub- and super-level sets components intersect in at most one component.

Contours

Problem: Given a point x, extract component of $f^{-1}(f(x))$ that contains x.

To extract the full contour, intersect every maximal simplex with the levelset.

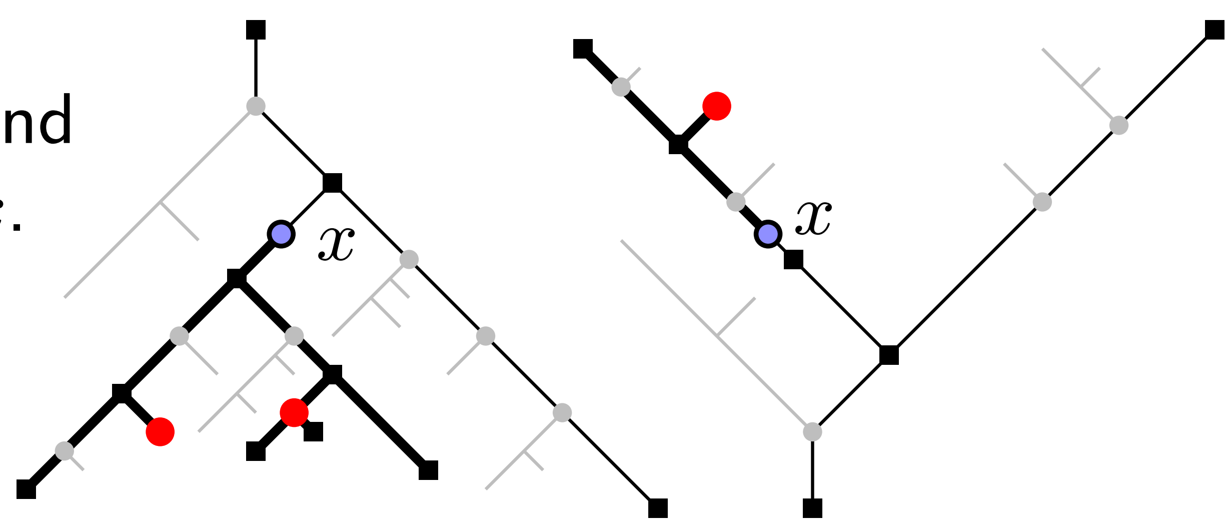
But we want to only report the component that contains x.



Idea: Local–global represenation determines a globally unique component ID without any communication. On simply connected domains, sub- and super-level sets components intersect in at most one component.

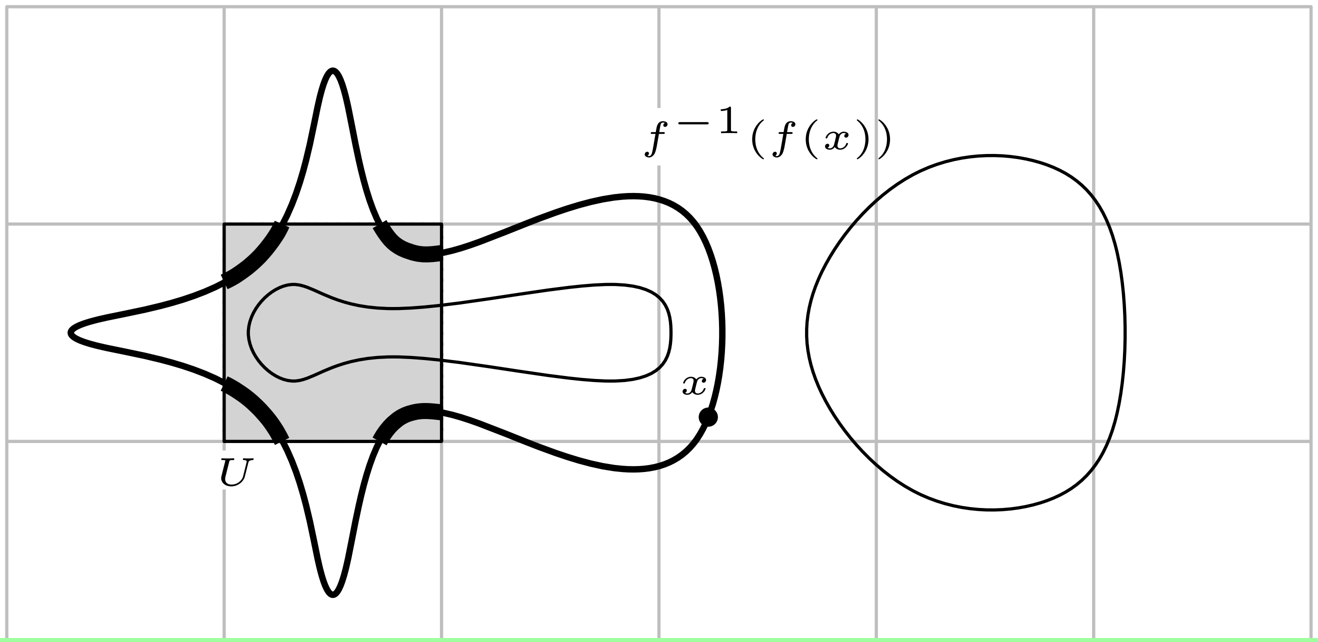
Algorithm:

- Processor responsible for x, identifies the minimum and the maximum of the sub- and super-levelset components that contain x.
- Each processor P_j identifies the sub- and super-levelset components containing x.
- Report only those simplices σ that have a vertex in each component.



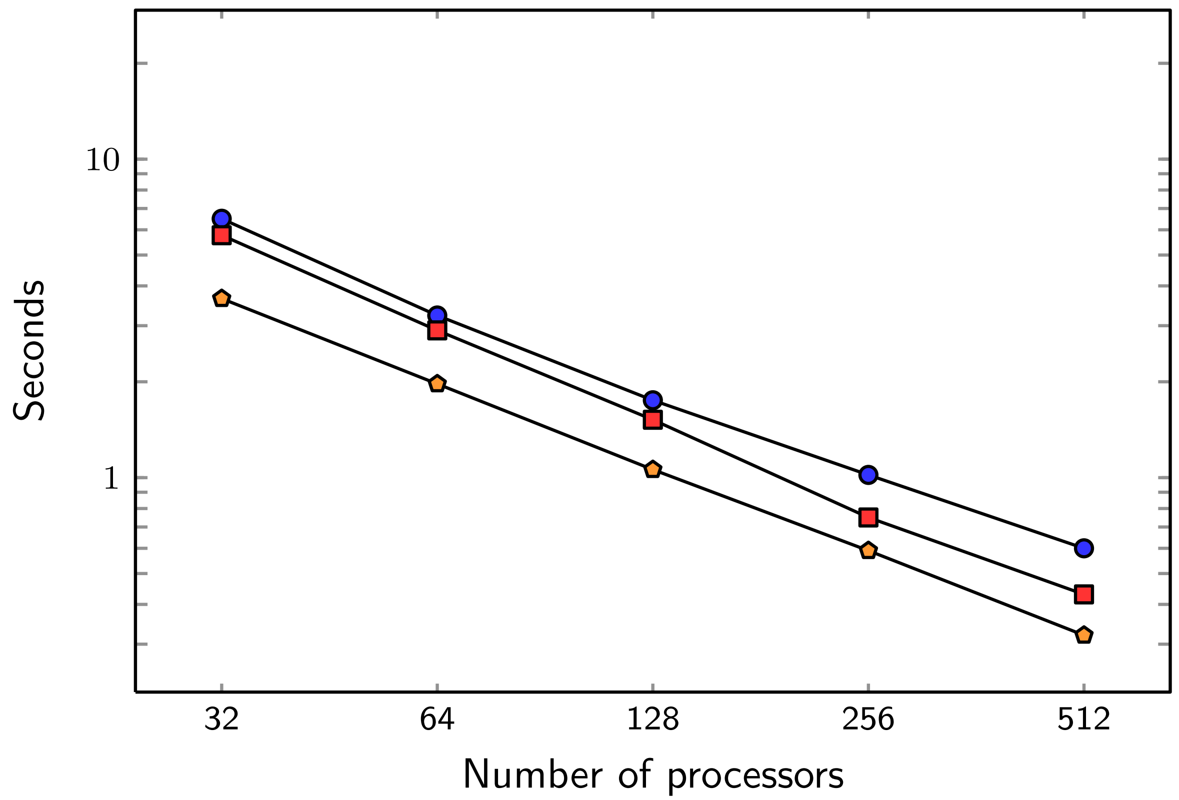
Analysis routine: levelset component extraction

Problem:
User chooses a point x,
extract component of
$f^{-1}(f(x))$ that contains x.



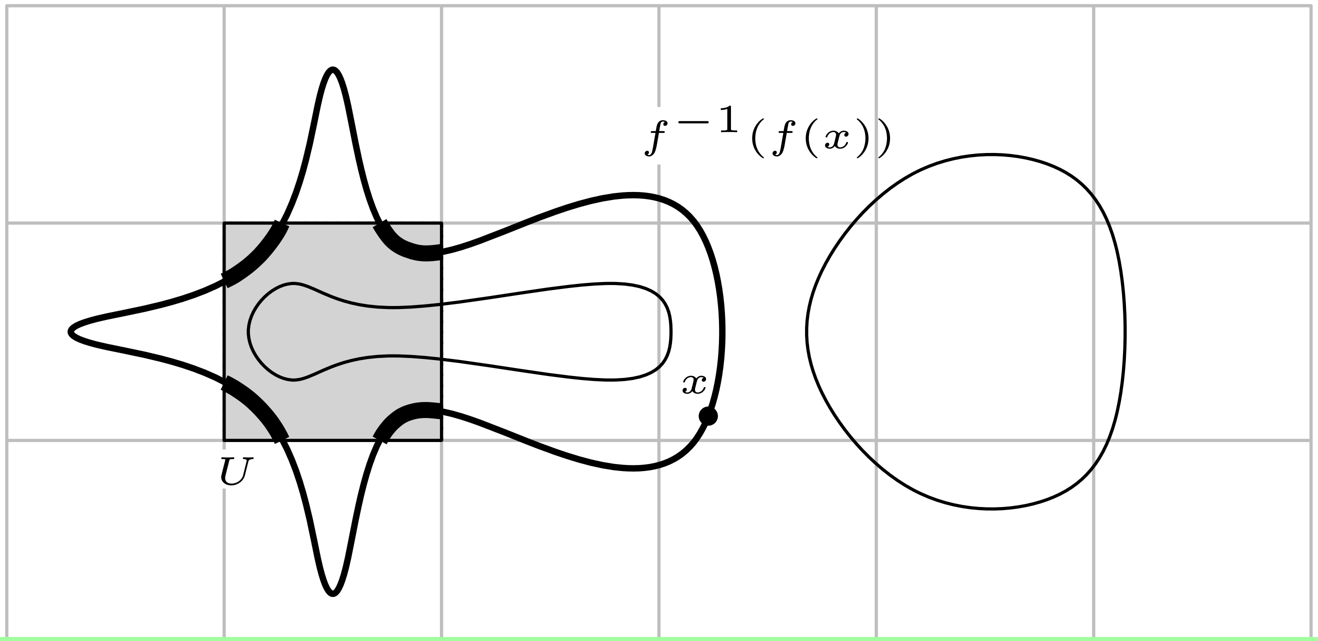
With local–global representation, this problem can be solved **without communication**:
each processor finds its contribution to the component; sufficient to broadcast just two vertices.

Result:
Input: 512^3 grids, medical images
Using local–global representation



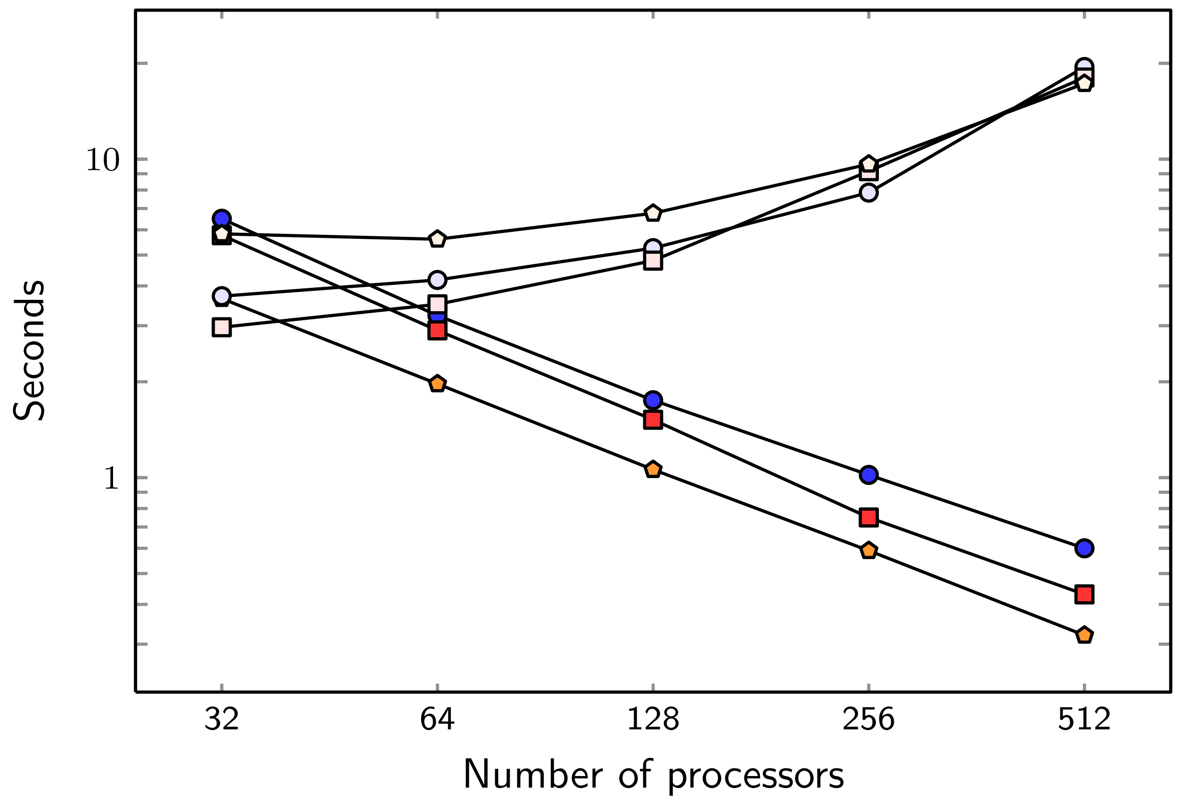
Analysis routine: levelset component extraction

Problem:
User chooses a point x,
extract component of
$f^{-1}(f(x))$ that contains x.



With local–global representation, this problem can be solved **without communication**:
each processor finds its contribution to the component; sufficient to broadcast just two vertices.

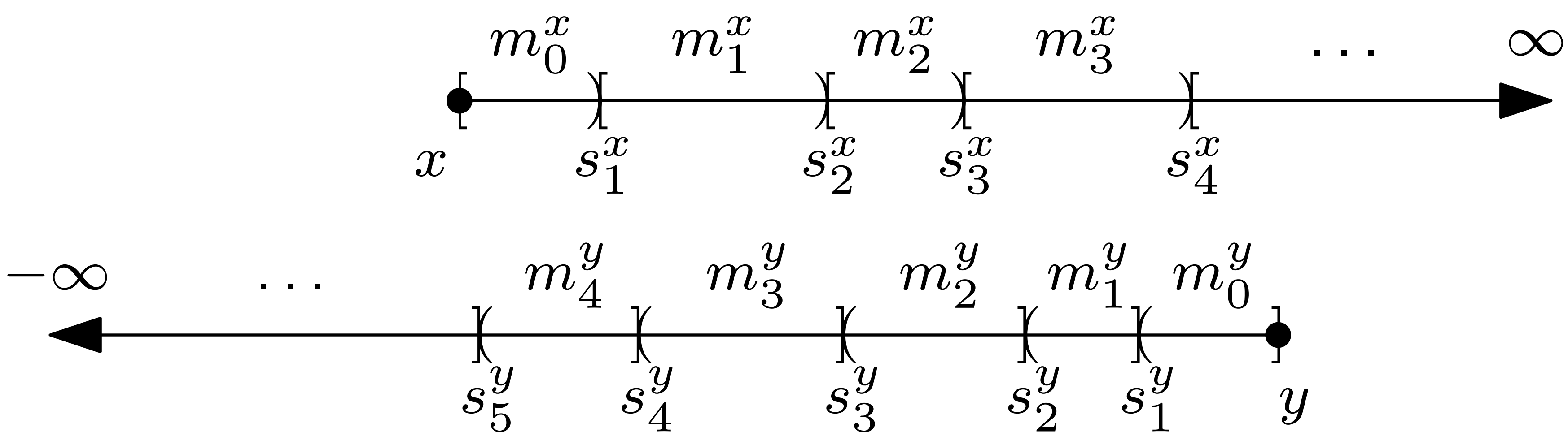
Result:
Input: 512^3 grids, medical images
Using local–global representation
vs. VisIt (state of the art)



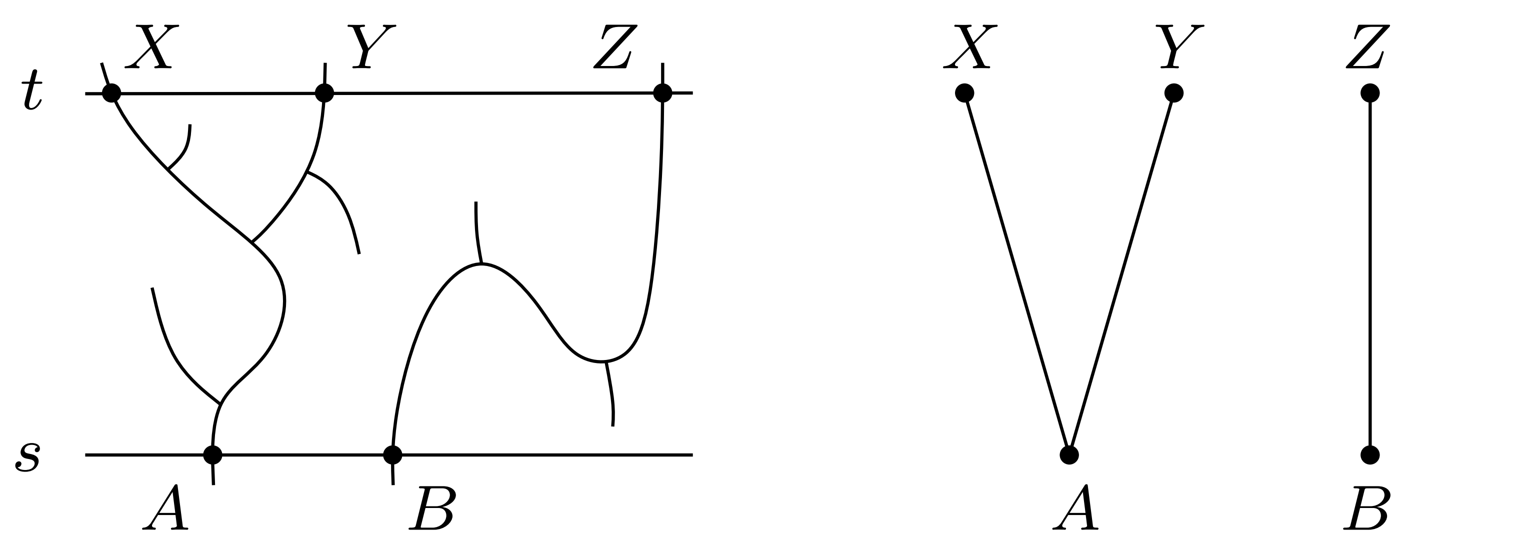
Variations

- **Component labeling:** instead of extracting a specific component, extract the full levelset, and label its components. We can do so without communication.

- **Interlevel set:** extract a branch or a path (x, y) in the contour tree.

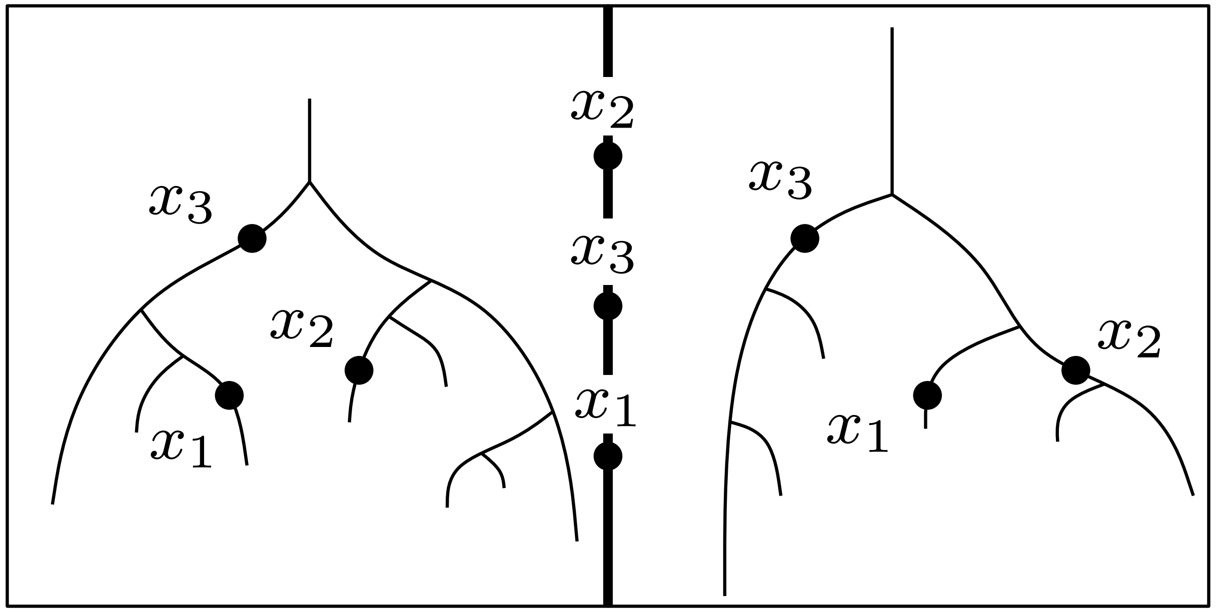


- **Contour tracking:** match contours of $f^{-1}(s)$ with those of $f^{-1}(t)$.



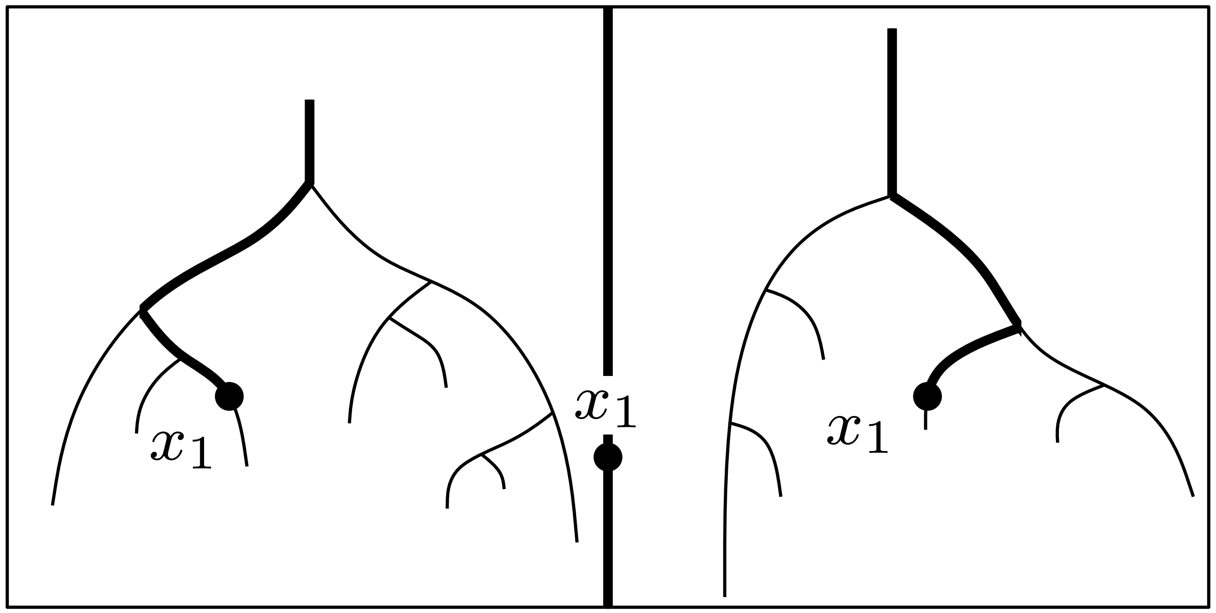
Shared-memory merging

- The basic operation in all three algorithm is the merging of two trees; this is done by repeating the union–find algorithm on the union of the two trees.
- We would like to take advantage of multiple shared-memory cores, but this procedure requires the vertices to be processed in the order of the function value.
- There is an alternative algorithm [Bremer et al.] that merges in sorted order the paths in the two trees that start from the shared vertices. (Unfortunately, this algorithm is much slower in the serial case than union-find.)



Shared-memory merging

- The basic operation in all three algorithm is the merging of two trees; this is done by repeating the union–find algorithm on the union of the two trees.
- We would like to take advantage of multiple shared-memory cores, but this procedure requires the vertices to be processed in the order of the function value.
- There is an alternative algorithm [Bremer et al.] that merges in sorted order the paths in the two trees that start from the shared vertices. (Unfortunately, this algorithm is much slower in the serial case than union-find.)

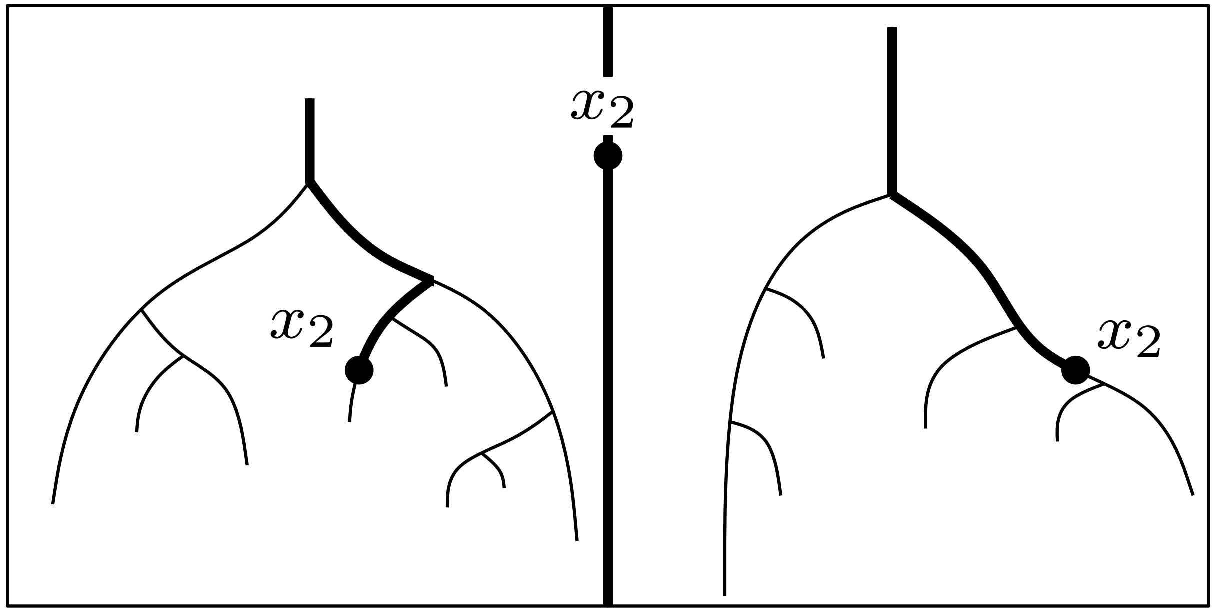


Shared-memory merging

- The basic operation in all three algorithm is the merging of two trees; this is done by repeating the union–find algorithm on the union of the two trees.

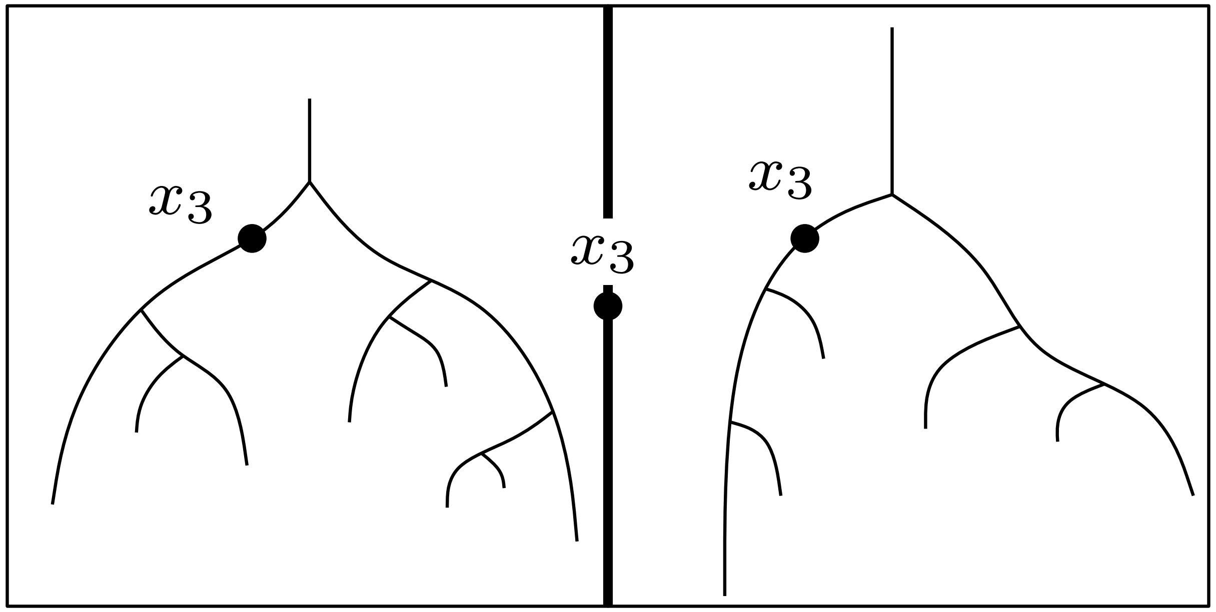
- We would like to take advantage of multiple shared-memory cores, but this procedure requires the vertices to be processed in the order of the function value.

- There is an alternative algorithm [Bremer et al.] that merges in sorted order the paths in the two trees that start from the shared vertices. (Unfortunately, this algorithm is much slower in the serial case than union-find.)



Shared-memory merging

- The basic operation in all three algorithm is the merging of two trees; this is done by repeating the union–find algorithm on the union of the two trees.
- We would like to take advantage of multiple shared-memory cores, but this procedure requires the vertices to be processed in the order of the function value.
- There is an alternative algorithm [Bremer et al.] that merges in sorted order the paths in the two trees that start from the shared vertices. (Unfortunately, this algorithm is much slower in the serial case than union-find.)

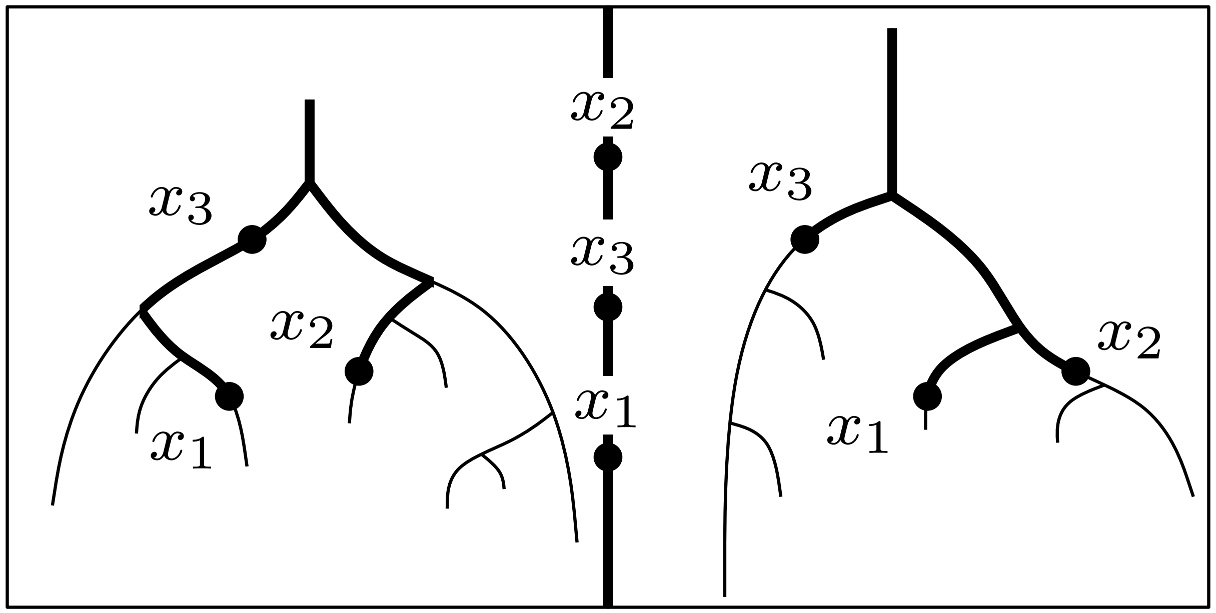


Shared-memory merging

- The basic operation in all three algorithm is the merging of two trees; this is done by repeating the union–find algorithm on the union of the two trees.

- We would like to take advantage of multiple shared-memory cores, but this procedure requires the vertices to be processed in the order of the function value.

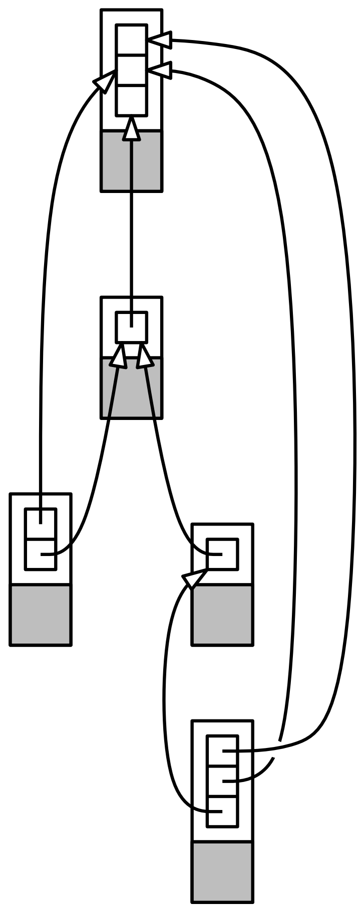
- There is an alternative algorithm [Bremer et al.] that merges in sorted order the paths in the two trees that start from the shared vertices. (Unfortunately, this algorithm is much slower in the serial case than union-find.)



Shared-memory merging

- The problem is that some vertices get traversed many more times than is necessary. (Merging n linked lists of size 1 each can take between $n \log n$ and n^2 depending on the chosen order. We don't control the order.)

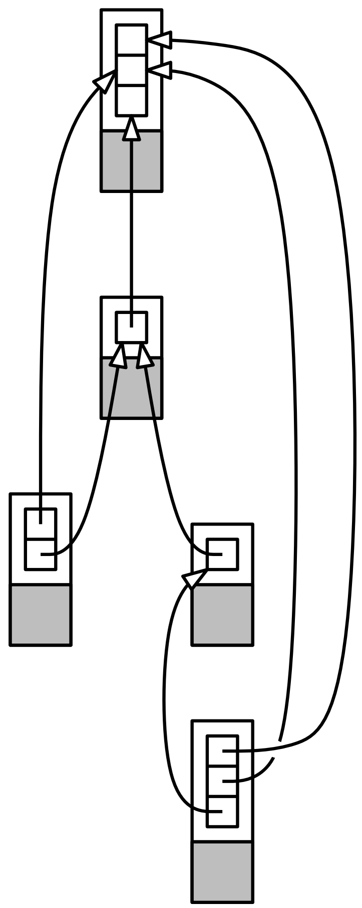
- Instead we turn to skip-lists (and build skip-trees):
  - Each parent pointer becomes a stack of randomized height;
  - Each path to the root is a skip-list;
  - When merging two skip-lists, we can use additional levels to skip over many nodes.

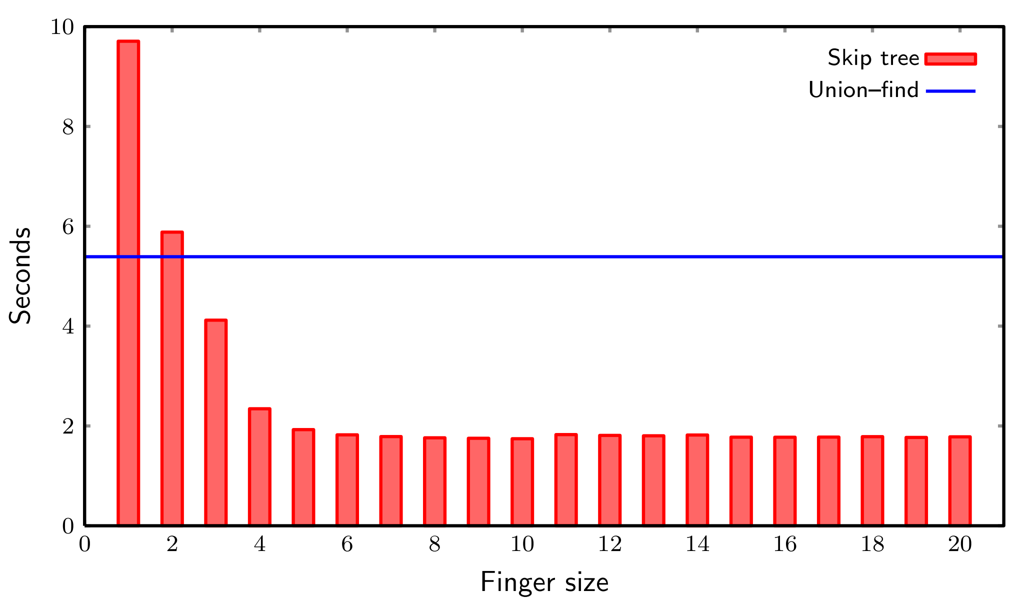


Shared-memory merging

- The problem is that some vertices get traversed many more times than is necessary. (Merging n linked lists of size 1 each can take between $n \log n$ and n^2 depending on the chosen order. We don't control the order.)

- Instead we turn to skip-lists (and build skip-trees):
  - Each parent pointer becomes a stack of randomized height;
  - Each path to the root is a skip-list;
  - When merging two skip-lists, we can use additional levels to skip over many nodes.



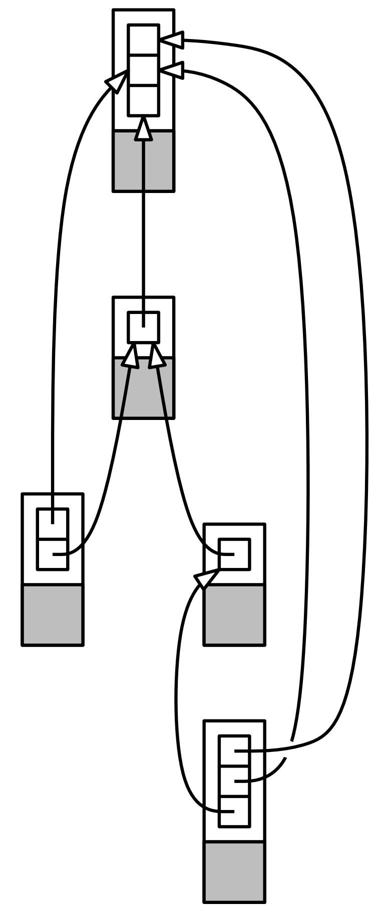


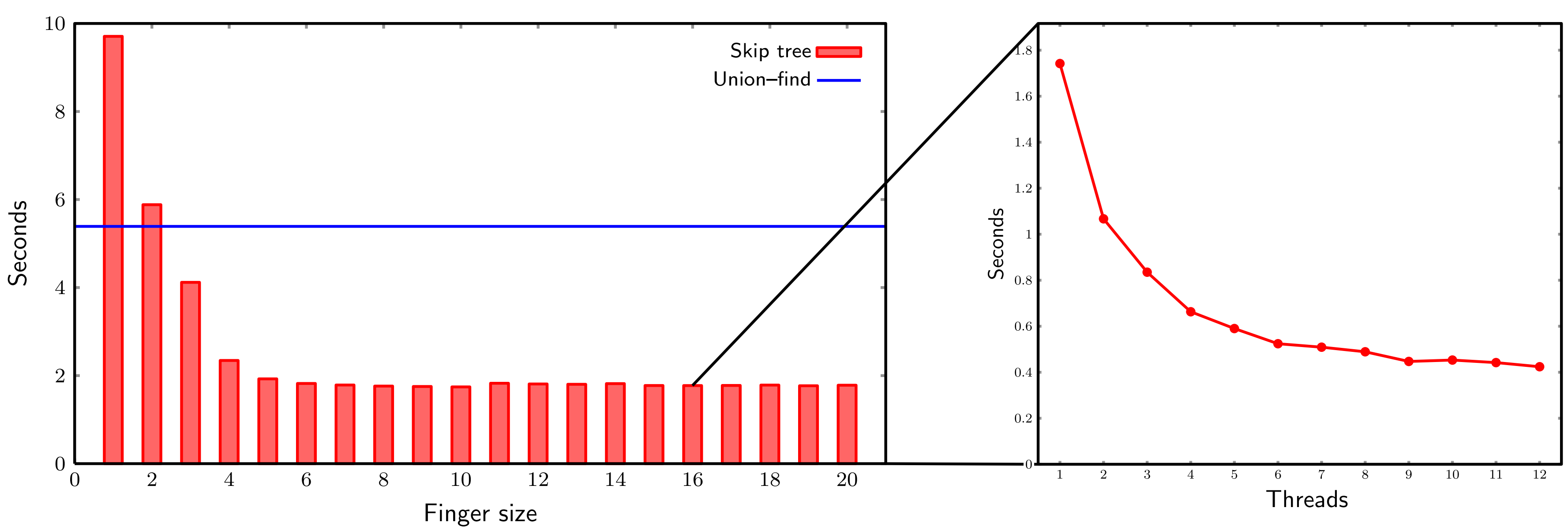
(Merging two trees with 800,000 nodes each.)

Shared-memory merging

- The problem is that some vertices get traversed many more times than is necessary. (Merging n linked lists of size 1 each can take between $n \log n$ and n^2 depending on the chosen order. We don't control the order.)

- Instead we turn to skip-lists (and build skip-trees):
  - Each parent pointer becomes a stack of randomized height;
  - Each path to the root is a skip-list;
  - When merging two skip-lists, we can use additional levels to skip over many nodes.





(Merging two trees with 800,000 nodes each.)

Summary

- Interleaving distance between merge trees.
  Major question: can we compute it efficiently?

- Two new ways to compute merge trees in parallel:
  - Global simplified: take advantage of the problem structure to prune noise;
  - Local–global: distribute the tree to facilitate analysis.

  (Can construct a tree on billions of points. Tried up to $4,096^3$.)

- A new way to merge two trees in parallel in shared memory.

- The shift of emphasis from parallel computation of the descriptor to its distributed representation that facilitates subsequent analysis is likely to benefit other topological constructions (Reeb graphs, Morse–Smale complexes, etc.).

Thank you for your time and attention!

This work was supported by the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 through

- Scientific Data Management and Analysis at Extreme Scale (DE-FOA-0000256)
- Use of resources at the National Energy Research Scientific Computing (NERSC) Center